AD 745022

A GENERALIZATION OF THE LR ALGORITHM TO SOLVE $AX = \lambda BX$

BY

LINDA KAUFMAN

STAN-CS-72-276

APRIL 1972

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

D D C
JUL 17 1972
C

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Stanford University
Computer Science Department
Stanford University (Stanford, California 94305)

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

A GENERALIZATION OF THE LR ALGORITHM TO SOLVE $AX = \lambda BX$

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report, April 1972

5. AUTHOR(S) (First name, middle initial, last name)

Linda Kaufman

6. REPORT DATE

April 1972

7a. TOTAL NO. OF PAGES

73

7b. NO. OF REFS

19

8a. CONTRACT OR GRANT NO

N00014-67-A-0112-00029

b. PROJECT NO

NR044-211

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

STAN-CS-72-276

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Releasable without limitations or dissemination.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Office of Naval Research
Arlington, Virginia 22217

13. ABSTRACT

In this paper, we will present and analyze an algorithm for finding x and λ such that $Ax = \lambda Bx$, where A and B are $n \times n$ matrices. The algorithm does not require matrix inversion, and may be used when either or both matrices are singular. Our method is a generalization of Rutishauser's LR method for the standard eigenvalue problem $Ax = \lambda x$ and closely resembles the QZ algorithm given by Moler and Stewart for the generalized problem given above. Unlike the QZ algorithm, which uses orthogonal transformations, our method, the LZ algorithm, uses elementary transformations. When either A or B is complex, our method should be more efficient.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

UNCLASSIFIED

Security Classification

A GENERALIZATION OF THE LR ALGORITHM TO SOLVE $AX = \lambda BX$

BY

Linda Kaufman

Abstract

In this paper, we will present and analyze an algorithm for finding $\underset{\sim}{x}$ and λ such that

$$A\underset{\sim}{x} = \lambda B\underset{\sim}{x} \tag{1}$$

where A and B are $n \times n$ matrices. The algorithm does not require matrix inversion, and may be used when either or both matrices are singular. Our method is a generalization of Rutishauser's LR method [17] for the standard eigenvalue problem $A\underset{\sim}{x} = \lambda\underset{\sim}{x}$ and closely resembles the QZ algorithm given by Moler and Stewart [10] for the generalized problem given above. Unlike the QZ algorithm, which uses orthogonal transformations, our method, the LZ algorithm, uses elementary transformations. When either A or B is complex, our method should be more efficient.

This research was supported in part by the National Science Foundation under grant number GJ 29988X and the Office of Naval Research under contract number N00014-67-A-0112-00029 NR 044-211. Reproduction in whole or in part is permitted for any purpose of the United States Government.

The LZ algorithm is based on three observations:

1) If L and M are nonsingular matrices, the eigenvalue problems $LAM\underset{\sim}{y} = \lambda LBM\underset{\sim}{y}$ and $A\underset{\sim}{x} = \lambda B\underset{\sim}{x}$ have the same eigenvalues and their eigenvectors are related by $\underset{\sim}{x} = M\underset{\sim}{y}$.

2) If A is a triangular matrix with diagonal elements α_i, and B is a triangular matrix with diagonal elements β_i, then for each i, $i = 1,2,---,n$, α_i/β_i is an eigenvalue of the generalized problem if $\beta_i \neq 0$. If for some i, β_i is zero, then the polynomial, determinant $(A-\lambda B)$, is of degree less than n. If α_i is not zero and the corresponding β_i is zero, we say that infinity is an eigenvalue. If for some i, both α_i and β_i are zero, then $\det(A-\lambda B)$ vanishes for all values of λ, and every scalar is an eigenvalue of $A\underset{\sim}{x} = \lambda B\underset{\sim}{x}$.

3) There exist matrices L and M such that LAM and LBM are upper triangular and L and M are the products of lower triangular and permutation matrices.

The first two observations should be obvious; the third requires explanation. In [18] Stewart shows that there exist two unitary matrices U and V such that

$$A' = U^H AV \quad \text{and} \quad B' = U^H BV$$

are upper triangular. The symbol U^H indicates the conjugate transpose of the matrix U. We can certainly write

$$U^H \text{ as } RL \quad \text{and} \quad V \text{ as } MS$$

where S and R are both upper triangular matrices and L and M are products of lower triangular and permutation matrices. The matrices $R^{-1}A'S^{-1} = LAM$ and $R^{-1}B'S^{-1} = LBM$ are both upper triangular and hence verify our observation.

The LZ algorithm has three parts. In the first part, the matrix A is reduced to upper Hessenberg form, i.e. $a_{ij} = 0$ for $i > j+1$, and B is simultaneously reduced to triangular form. The first stage of the LZ algorithm is very similar to the first stage of the Moler-Stewart algorithm, and they may be freely substituted for each other. The advantage of using our method is that it is about 5/2 faster; the advantage of theirs is numerical stability. The second stage of the LZ algorithm is a generalization of the LR algorithm and iteratively reduces A to triangular form while preserving the triangularity of B. The last part of LZ obtains the eigenvectors of the triangular matrices and transforms them back into the original coordinate system. Throughout the algorithm stabilized elementary transformations (see Wilkinson [19], p. 164) are used to insure numerical stability. These transformations are the products of lower triangular matrices and permutation matrices, and are easy to compute and easy to use. The permutation matrices are designed to help minimize the loss of accuracy in numerical operations. A further explanation of the stabilized elementary transformations used in the heart of the LZ algorithm is contained in the notation section at the end of this introduction.

I. BACKGROUND

As Lancaster [8] and Gantmacher [6] point out, the generalized eigenvalue problem often occurs in the physical sciences. Many mechanical and electrical systems are governed by a differential equation of the form

$$C\ddot{x} + D\dot{x} + Ex = 0$$

where C, D, and E are $n \times n$ matrices and the solution is expected to have the form $x(t) = e^{\lambda t}x(0)$. Solving the ordinary differential equation entails finding the eigensystem of

$$(\lambda^2 C + \lambda D + E)\underline{x} = 0$$

If no damping occurs and the D term is missing, the problem is like the one given in (1) in λ^2. If the system is damped and the D term is present, the problem can be reconstructed to have the form of (1) where now A is the matrix

$$\begin{bmatrix} D & I \\ C & 0 \end{bmatrix}$$

and B is the matrix

$$\begin{bmatrix} -E & 0 \\ 0 & I \end{bmatrix}$$

In many problems, especially those which describe physical systems, A and B have some special structure and most of the algorithms in the literature are designed for matrices having specific properties. In [9] Martin and Wilkinson have given a method for A,B symmetric and B positive definite. Crawford [2] has presented a modification of that algorithm when B is a band matrix. In [7], Golub, Underwood and Wilkinson describe a version of the Lanczos algorithm for A,B symmetric and B positive definite. Fix and Heiberger [3] have a method designed for illconditioned B which requires the determination of the rank of certain submatrices in A and B. If symmetry and positive definiteness are not present and B is well conditioned, the eigensystem of $Ax = \lambda Bx$ can be found by forming $B^{-1}A$ and determining the eigensystem of $B^{-1}Ax = \lambda x$, for which there exist several good methods. For a nearly singular B, Peters and Wilkinson [15]

describe an algorithm which approximates the null space of B. This approach involves determining the rank of submatrices which is often difficult to do exactly on a finite precision computer.

Recently Moler and Stewart [10] have presented an algorithm for solving the generalized eigenvalue problem which may be used regardless of the condition and structure of the two matrices. Our algorithm resembles the QZ algorithm in that we generalize Rutishauser's LR algorithm [17] in the same way that Moler and Stewart generalize Francis's QR method [5] for the standard eigenvalue problem $Cx = \lambda x$. Before we describe our algorithm in detail and discuss its relationship to the QZ method perhaps it is best to review the QR and the LR methods. In practice, the <u>LR method</u> for the problem $Cx = \lambda x$ is essentially:

1) Reduce C to upper Hessenberg form using similarity transformations.

2) Determine a shift λ.

3) Find L, a product of stabilized elementary transformations, and R, an upper triangular matrix, so that $L(C - \lambda I) = R$.

4) Set C to LCL^{-1}. The matrix C will be upper Hessenberg.

5) If the subdiagonal elements of C are not negligible, return to 2.

6) The eigenvalues of the original matrix are the diagonal elements of C.

The shift λ is used to speed up the algorithm, which converges according to the ratios of the shifted eigenvalues. In practice, the shift i usually an eigenvalue of the lowest 2 x 2 subblock on the diagonal of C which has not been triangularized. This policy often gives a good

approximation to an eigenvalue of the whole matrix.

The basic QR method is approximately the same as the LR method with an orthogonal matrix Q replacing the matrix L in steps 3 and 4. In practice a double shift implicit version of the QR method is used in which steps 3 and 4 read

3) Find Q, an orthogonal matrix, and R, an upper triangular matrix, such that

$$R = QT \equiv Q(C - \lambda I)\ (C - \bar{\lambda} I) \text{ where } \lambda \text{ and } \bar{\lambda}$$

are complex conjugate shifts or a pair of real shifts.

4) Set C to QCQ^T.

Only the first column of T is ever explicitly formed.

The main advantage of the double shift algorithm is the preservation of real arithmetic for real matrices. The QZ algorithm also has this property. With the double step QR and QZ methods the final matrix is not necessarily triangular, but may have 2 x 2 submatrices on the diagonal which must be resolved. The LR and the LZ algorithms do not limit themselves to real arithmetic but avoid the 2 x 2 problem. Double shift LR and LZ methods are not found in practice because of the lack of a theoretical basis. Francis [5] has proved that one iteration of the implicit double shift QR method is equivalent to two iterations of the basic QR method. His theorem is based on the uniqueness of orthogonal transformations which reduce a given matrix to triangular form with positive diagonal elements. This uniqueness property is missing for stabilized elementary transformations.

II. The LZ Algorithm as a Generalization of the LR Algorithm

The LZ algorithm is motivated by the LR method described above where the matrix C is AB^{-1}. However, we do not assume that B^{-1} exists. Briefly, our algorithm is:

1) Reduce A to upper Hessenberg form and B to triangular form.

2) Find a shift λ.

3) Find matrices L and M, stabilized elementary transformations, such that $L(A - \lambda B)$ is upper triangular and LBM is upper triangular.

4) Set A to LAM and B to LBM. The new A will be upper Hessenberg.

5) If the subdiagonal elements of A are not negligible, return to 2.

6) The ith eigenvalue is a_{ii}/b_{ii} if b_{ii} is nonzero.

Again the shift is used to hasten the convergence of the algorithm. In practice it is usually a solution of the lowest 2×2 subproblem on the diagonal of $A - \lambda B$ which has not been triangularized.

If the matrix C in the LR method is AB^{-1}, then the matrix L in the third step of the LR method is precisely the matrix L in the third step of the LZ method if both algorithms employ the same pivoting strategy. This fact is verified by denoting the left hand transformation in the third step of the LZ algorithm by $\bar{L}$ and noticing that

$$\begin{aligned}\bar{L}(C - \lambda I) &= \bar{L}(AB^{-1} - \lambda I)\\ &= \bar{L}(A - \lambda B)B^{-1}\end{aligned}$$

an upper triangular matrix. Thus $\bar{L}$ is also a transformation which

triangularizes $C - \lambda I$; and if the pivoting strategy is the same, $\bar{L}$ is the transformation L in step 3 of the method LR.

Moreover, it can be shown that the two algorithms produce corresponding iterates. If C' denotes the next iterate in the LR algorithm and A' and B' are the iterates in the LZ algorithm, then in the LR algorithm

$$C' = LCL^{-1}$$

and in the LZ algorithm

$$\begin{aligned} A'B'^{-1} &= LAMM^{-1}B^{-1}L^{-1} \\ &= LAB^{-1}L^{-1} \\ &= LCL^{-1} \end{aligned}$$

In Chapter One we will present two algorithms. The first is a straight generalization of the LR method. The second is an implicit scheme in which only the first column of $A - \lambda B$ is actually formed. The second scheme requires fewer operations and is more stable.

III. NOTATION

To simplify the explanation in the remainder of this paper, we introduce the following symbols:

For a complex scalar α, $\|\alpha\|$ will denote $|\mathrm{Im}(\alpha)| + |\mathrm{Re}(\alpha)|$. $\|\alpha\|$ corresponds to the 1 norm of α, if α is considered as a vector in the complex plane.

In general, the $(i,j)^{th}$ element of the matrix A will be denoted by a_{ij}. If a matrix A is the kth element in a sequence of matrices, it will be designated by A_k and its (i,j) element will be designated by $a_{ij}^{(k)}$.

$\mathcal{L}_i$ will denote the subset of the set of stabilized elementary matrices described in Wilkinson [19] having the form

$$\begin{array}{cc} & \overbrace{\qquad\qquad\qquad}^{i^{th}\text{ column}} \\ i^{th}\text{ row} \rightarrow & \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & \eta_i & 1 & \\ & & & & 1 \end{bmatrix} \end{array} ,$$

where $\|\eta_i\| \leq 1$, or the form

$$\begin{array}{cc} & \overbrace{\qquad\qquad\qquad}^{i^{th}\text{ column}} \\ i^{th}\text{ row} \rightarrow & \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 0 & 1 & \\ & & 1 & \eta_i & \\ & & & & 1 \end{bmatrix} \end{array} ,$$

where $\|\eta_i\| < 1$. Blank spaces indicate zeroes.

When a matrix is multiplied on the left by a matrix of $\mathcal{L}_i$ of the first form only its $i+1^{st}$ row is changed, but when a matrix is multiplied on the left by a matrix of $\mathcal{L}_i$ of the second form, its ith and $i+1^{st}$ rows are first interchanged and then a multiple of the new i^{th} row is added to the $i+1^{st}$ row.

We will often use a member of $\mathcal{L}_i$ to annihilate an element in the $i+1^{st}$ row of a matrix. For example, we may want to zero $a_{i+1,j}$. If either the current $a_{i+1,j}$ or a_{ij} is nonzero, then there exists a unique member of $\mathcal{L}_i$ which will annihilate $a_{i+1,j}$. Specifically, if $\|a_{i+1,j}\|$ is less than or equal to $\|a_{ij}\|$, we use a matrix of the first form where η_i is $-a_{i+1,j}/a_{ij}$; otherwise, we use a matrix of the second form with η_i given by $-a_{ij}/a_{i+1,j}$. If both the current $a_{i+1,j}$ and a_{ij} are zero, then any member of $\mathcal{L}_i$ will leave a zero in $a_{i+1,j}$.

$\mathcal{M}_i$ will denote the subset of stabilized matrices having the form

$$\begin{array}{cc} & \overbrace{\qquad\qquad}^{i^{th}\text{ column}} \\ i^{th}\text{ row} \longrightarrow & \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & \mu_i & 1 & \\ & & & & 1 \end{bmatrix} \end{array}$$

where $\|\mu_i\| \leq 1$, or the form

$$\begin{array}{cc} & \overbrace{\qquad\qquad}^{i^{th}\text{ column}} \\ i^{th}\text{ row} \longrightarrow & \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \mu_i & 1 & \\ & & 1 & 0 & \\ & & & & 1 \end{bmatrix} \end{array},$$

where $\|\mu_i\| < 1$.

When a matrix is multiplied on the right by a matrix of the first form, only its ith column is changed, but when a matrix is multiplied on the right by a matrix of the second form, its ith and $i+1^{st}$ columns are interchanged and a multiple of the new $i+1^{st}$ column is added to the ith column.

The set $\mathcal{T}$ will denote the set of matrices in upper triangular form. If A is in $\mathcal{T}$, then $a_{i,j} = 0$ for $i > j$. The set $\mathcal{H}$ will denote the set of matrices in upper Hessenberg form. If A is in $\mathcal{H}$, then $a_{ij} = 0$ for $i > j+1$.

Each iteration of the LZ algorithm involves multiplying matrices by product of transformations. In our discussion of the LZ algorithm the symbol A' will usually denote the matrix A after all the transformations for one iteration have been applied to it. The symbol A^* will represent the matrix A after some but not all of the transformations for one iteration have been applied to it.

CHAPTER ONE

In this chapter we shall describe the LZ algorithm in detail. As mentioned in the introduction, the algorithm has 3 sections:

1) Reducing B to triangular form and transforming A to upper Hessenberg form.

2) Iteratively reducing A to triangular form while preserving the triangularity of B.

3) Finding the eigenvectors of the triangular system and transforming them into the eigenvectors of the original system.

To obtain the eigenvectors of the original system all the right hand transformations must be accumulated, for if L and M are nonsingular matrices and

$$LAM\underset{\sim}{y} = \lambda LBM\underset{\sim}{y}$$

then

$$A\underset{\sim}{x} = \lambda B\underset{\sim}{x}$$

where $\underset{\sim}{x} = M\underset{\sim}{y}$. Thus, if $\underset{\sim}{y}$ is an eigenvector of the triangular system, $M\underset{\sim}{y}$ is an eigenvector of the original system.

In the second section of this chapter, where we present the iterative section of the algorithm, we will describe two algorithms. The first method is an explicit scheme which, if B^{-1} existed, would be quite like the LR algorithm for AB^{-1}. The second algorithm is a more stable implicit scheme. In the third section we prove that the two algorithms generate and use the same transformations.

I. INITIAL REDUCTION TO HESSENBERG-TRIANGULAR FORM

In this section an algorithm will be described that reduces a matrix A to upper Hessenberg form and reduces a matrix B to triangular form by applying the same elementary transformations to both matrices.

The first step is the standard Gaussian elimination process with partial pivoting as described in Forsythe and Moler [4]. We find a matrix L, the product of elementary and permutation matrices such that LB is upper triangular, and then replace A and B by LA and LB, respectively.

In the next stage we reduce A to an element of $\mathcal{H}$ while maintaining the triangularity of B. We begin by choosing an element L_{n-1} from $\mathcal{L}_{n-1}$ so that replacing A by $L_{n-1}A$ puts a zero in the (n,1) position of A. Multiplying B on the left by L_{n-1} introduces a new nonzero element in the (n,n-1) position of B. If pivoting had been necessary, B would still have the same form. Thus A and B now look like

A	B
X X X X X	X X X X X
X X X X X	0 X X X X
X X X X X	0 0 X X X
X X X X X	0 0 0 X X
0 X X X X	0 0 0 X X

We now focus on B, and choose a matrix M_{n-1} from $\mathcal{M}_{n-1}$ so that setting B to BM_{n-1} and A to AM_{n-1} returns B to triangular form and maintains the zero we introduced into A. Thus we have

$$
\begin{array}{ccc}
A & & B \\
\begin{matrix} X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \end{matrix} & & \begin{matrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{matrix}
\end{array}
$$

The process is continued until A is in $\mathcal{N}$. As each element in A is zeroed using a row transformation, a nonzero element is introduced on the subdiagonal of B which must be immediately annihilated using a column transformation. Elements are eliminated from A in the order given below:

$$
\begin{matrix}
X & X & X & X & X \\
X & X & X & X & X \\
X^{3} & X & X & X & X \\
X^{2} & X^{5} & X & X & X \\
X^{1} & X^{4} & X^{6} & X & X
\end{matrix}
$$

Note that pivoting maintains stability but does not affect the zero structure of the two matrices any more than a nonpivoting algorithm would.

There are other ways to annihilate elements of A and B which might be more efficient or more stable for any given problem. One such method involves reducing to B to an element of $\mathcal{T}$ and then using column transformations to reduce A to an element of $\mathcal{N}$. The nonzero elements, which are introduced on the subdiagonal of B by the column transformations, are eliminated using row transformations. Elements of A would be zeroed in the order given below:

$$\begin{matrix} X & X & X & X & X \\ X & X & X & X & X \\ X^6 & X & X & X & X \\ X^4 & X^5 & X & X & X \\ X^1 & X^2 & X^3 & X & X \end{matrix}$$

This algorithm would certainly be more efficient than the first method described if B were the identity matrix and if A were the matrix

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Both algorithms just described require about $13n^3/6$ multiplications and $13n^3/6$ additions. In terms of the first method the operation count can be broken down in the following way:

	Additions + Multiplications
1) Reducing B to triangular form	
Transformations on A	$n^3/2$
Transformations on B	$n^3/3$
2) Reducing A to an element of $\mathcal{N}$ and preserving the triangularity of B	
a) To eliminate elements in the jth column of A	
Transformations on A	$(2n-j)\ (n-i-j)$
Transformations on B	$(n+2)\ (n-1-j)$

b) Total 2nd step

Transformations on A	$5n^3/6$
Transformations on B	$n^3/2$

If the eigenvectors are requested, then, as we explained at the beginning of this chapter, the product of the M's must be accumulated. This requires $n^3/2$ additions and $n^3/2$ multiplications.

In comparison, the first part of the QZ algorithm requires $17n^3/3$ multiplications, $17n^3/3$ additions, and n^2 square roots. If eigenvectors are also desired, the QZ algorithm expends an additional $3n^3/2$ multiplications and $3n^3/2$ additions.

It is interesting to compare the above figures with the number of operations needed to form AB^{-1} and reduce this matrix to Hessenberg form. If iterative refinement is not done, the basic process requires about $13n^3/6$ multiplications and $13n^3/6$ additions or the same number required for the first section of LZ. If eigenvectors are desired, $n^3/2$ extra multiplications and $n^3/2$ additions are needed. The figures in this paragraph assume nonunitary transformations are being used.

The following table summarizes the cost of using the initial part of the three algorithms.

Summary of Operation Counts

	Without Eigenvectors			With Eigenvectors		
	+	×	square roots	+	×	square roots
LZ	$13n^3/6$	$13n^3/6$	0	$16n^3/6$	$16n^3/6$	0
QZ	$34n^3/6$	$34n^3/6$	n^2	$43n^3/6$	$43n^3/6$	n^2
AB^{-1}	$13n^3/6$	$13n^3/6$	0	$16n^3/6$	$16n^3/6$	0

II. FINDING THE EIGENVALUES

In this section we give an algorithm for determining the eigenvalues of the problem $A\underset{\sim}{x} = \lambda B\underset{\sim}{x}$ where A is upper Hessenberg and B is upper triangular. As in Moler's and Stewart's QZ algorithm, the method entails iteratively reducing A to upper triangular form while preserving the triangularity of B.

Each iteration of the iterative procedure is essentially:

1) Find a shift λ which could be a_{nn}/b_{nn} or an eigenvalue of the lowest 2 by 2 subproblem of $A - \lambda B$.

2) Find a matrix L such that $L(A - \lambda B)$ is upper triangular.

3) Find a matrix M such that LBM is upper triangular.

4) Set A' to LAM and B' to LBM. A' will be in $\mathcal{H}$.

Most of this section discusses the construction of L and M and their application to our matrices to satisfy the requirements given above. If the matrix T denotes $A - \lambda B$, then it is obvious that $LAM = LTM + \lambda LBM$ and that LAM is in $\mathcal{H}$ if and only if LTM is in $\mathcal{H}$.

Each iteration begins with an A which has no zero subdiagonal elements. If after the iteration A' has a zero on its subdiagonal, we can deflate the problem and work on a lower dimensional subproblem. Hence the purpose of each iteration is to drive the elements on the subdiagonal A' closer to zero. In Chapter 2 we will specify conditions under which the process, we are about to describe, accomplishes this goal.

In the 'explicit' version of the algorithm, we start an iteration by forming $T = A - \lambda B$. Since A is upper Hessenberg and B is upper triangular, our matrices look like

$$T = \begin{matrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{matrix} \qquad B = \begin{matrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{matrix}$$

We now select an element L_1 from $\mathcal{L}_1$ so that replacing T by L_1T zeroes the element in the (2,1) position of T. Then replacing B by L_1B introduces a nonzero element in the (2,1) position of B. T and B now have the form

$$T = \begin{matrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{matrix} \qquad B = \begin{matrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{matrix}$$

Our attention is now turned to B, and M_1 is chosen from $\mathcal{M}_1$ so that replacing B by BM_1 yields a triangular matrix. The application of M_1 to T is delayed. We now return to T and annihilate t_{32} by an element L_2 in $\mathcal{L}_2$. Applying the same transformation to B produces

$$T = \begin{matrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{matrix} \qquad B = \begin{matrix} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{matrix}$$

The matrix M_1 is now applied to T and M_2 is chosen from $\mathcal{M}_2$ so that BM_2 is in $\mathcal{T}$. The matrices T and B now look like

T	B
X X X X X	X X X X X
X X X X X	0 X X X X
0 0 X X X	0 0 X X X
0 0 X X X	0 0 0 X X
0 0 0 X X	0 0 0 0 X

It is important to notice that the element t_{31} is zero. Future transformations will not touch this element, and hence it will remain zero throughout the iteration. Furthermore, the situation had not been influenced by the form of M_1, i.e. whether pivoting had been necessary to stably zero b_{21}.

In general, row transformations are applied to T and B simultaneously but column transformations are applied first to B. If we write M as $M_1M_2 \ldots M_{n-1}$, then as we apply M_i to B we apply M_{i-1} to T. Each row transformation will zero an element of T and introduce a nonzero on the subdiagonal of B. Similarly, each column transformation returns B to triangular form while introducing a new nonzero element on the subdiagonal of T. Delaying the application of the right transformations to T ensures us that the new nonzero element produced will not affect future row transformations on T, i.e., T will remain in $\mathcal{X}$. According to the law of association of multiplication of matrices the delay is legal. In summary, the <u>explicit algorithm</u> for each main iteration step is given by:

Set T to $A \to B$ for $i = 1, 2, \ldots, n-1$.

1) Find L_i to stably zero $t_{i+1,i}$ and set T to L_iT and B to L_iB.

2) If $i > 1$, set T to TM_{i-1}.

3) Find M_i to stably zero $b_{i+1,i}$ and set B to BM_i.

Set T to TM_{n-i}.

Set A to $T + \lambda I$.

As in the LR algorithm, the subdiagonal elements of A should become small and approach zero at a rate determined by the ratio of the eigenvalues. By using a shift λ, we hope to hasten the process.

There is only one major drawback to the algorithm just described: it is potentially unstable. If the shift λ is large relative to the size of the elements of A, information needed to find future small eigenvalues can be lost when T is explicitly formed. This will occur when the shift is computed from the lower 2 x 2 submatrix of AB^{-1} and much smaller elements appear in the bottom of B than in the top of B.

The following example indicates the deterioration that can occur with the explicit algorithm. The relative residual is the quantity

$$\|\beta_i A\underset{\sim}{x} - \alpha_i B\underset{\sim}{x}\|_\infty / (|\beta_i| \, \|A\|_\infty + |\alpha_i| \, \|B\|_\infty)$$

where β_i is the ith diagonal element of the final triangular matrix LBM and α_i is the ith diagonal element of the final triangular matrix LAM. The significance of this quantity is that we have really solved the problem $\beta(A+E)\underset{\sim}{x} = \alpha(B+F)\underset{\sim}{x}$. The ith eigenvalue is given by α_i/β_i. This problem was done on an IBM 360 machine in double precision.

$$\overset{A}{\begin{pmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 0.0 & 7.0 & 8.0 \end{pmatrix}} \quad \overset{B}{\begin{pmatrix} 1.0 & 10.0 & 15.0 \\ 0.0 & 1.*10^{-15} & 1.*10^{-5} \\ 0.0 & 0.0 & 1.*10^{-15} \end{pmatrix}}$$

Eigenvalue	Relative Residual
$-.6999919999436334*10^{21}$	$.7242069776452181*10^{-17}$
$-.1142870204249628*10^{6}$	$.8643463693672128*10^{-2}$
0.0	.2666666666666667

According to these results, zero is an eigenvalue of the problem, which contradicts the fact that A is nonsingular, and in two instances the relative residual is so large that their corresponding eigenvalues must solve a problem which cannot be considered close to the original problem.

The instability mentioned above can be avoided if T is never formed. We will now describe an implicit algorithm which works with A and B directly. When this new method was applied to the above example, the following results were obtained:

Eigenvalue	Relative Residual
$-.6999920003499999*10^{21}$	$.4644067519786 57*10^{-17}$
$-.1400015183156 42*10^{7}$	$.6036930705474 78*10^{-17}$
.183673576484812	$.1497153916764 53*10^{-16}$

The small relative residuals indicate that the eigenvalues solve a problem which is close to the original eigenvalue problem.

We note that with the standard eigenvalue problem $A\underset{\sim}{x} = \lambda \underset{\sim}{x}$, Gershgorin's theorem (19) assures us that computing the shift from the eigenvalues of the lower 2 by 2 of A will not give us a shift larger than the norm of A.

In our description of the new implicit algorithm transformations will be denoted by $\underline{L}_i$ and $\underline{M}_i$ but, as we shall prove, the implicit and explicit algorithms are essentially equivalent in the absence of roundoff error, and except in one instance, the $\underline{L}_i$'s of the implicit algorithm are the $\underline{L}_i$'s of the explicit algorithm. The same is true for the $\underline{M}_i$'s.

The implicit algorithm begins by forming the same L_1 that was formed in the explicit algorithm from a_{11}, b_{11}, a_{21} and λ. The matrix L_1 is applied to A and B and obviously the same nonzero element is introduced in the (2,1) position of B as in the explicit algorithm. M_1 is again formed so that BM_1 is in $\mathcal{T}$, but this time, M_1 is also applied to A. At this point A and B look like

A	B
X X X X X	X X X X X
X X X X X	0 X X X X
X X X X X	0 0 X X X
0 0 X X X	0 0 0 X X
0 0 0 X X	0 0 0 0 X

We now select $\underline{L}_2$ from $\mathcal{L}$, so that $\underline{L}_2 A$ is in $\mathcal{N}$. When $\underline{L}_2$ is applied to B, a new nonzero element is introduced in the (3,2) position of B which is then annihilated by $\underline{M}_2$.

In general, row transformations return A to upper Hessenberg form and introduce a nonzero element on the subdiagonal of B. Column transformations return B to upper triangular form and produce a nonzero element on the second subdiagonal of A. In contrast to what occurs in the explicit algorithm, in the implicit method, column transformations are applied simultaneously to both matrices. In more detail each iteration of the <u>implicit algorithm</u> is given by:

1) Set γ to $a_{11} - \lambda b_{11}$ and δ to a_{21}.

2) If $|\delta| > |\gamma|$, L_1 is the element of $\mathcal{L}_1$ using pivoting with $\eta_1 = -\gamma/\delta$; otherwise L_1 is the element of $\mathcal{L}_1$ without pivoting with $\eta_1 = -\delta/\gamma$. Set A to L_1A and B to L_1B. Set i to 1.

3) Find $\underline{M}_i$, an element of $\mathcal{M}_i$, to stably zero $b_{i+1,i}$ and set A to $A\underline{M}_i$ and B to $B\underline{M}_i$. If $i = n-1$, stop.

4) Set i to i+1. Find $\underline{L}_i$, an element of $\mathcal{L}_i$, to stably zero $a_{i+1,i-1}$ and set A to $\underline{L}_iA$ and B to $\underline{L}_iB$. Return to 3.

For the implicit method, about $2n^2$ multiplications and $2n^2$ additions are required per iteration. If eigenvectors are also requested, n^2 multiplications and n^2 additions must be spent to accumulate the M's. In contrast, the QZ algorithm requires $13n^2$ additions, $13n^2$ multiplications and 3n square roots per iteration, and $8n^2$ additional multiplications and additions if eigenvectors are requested. However, it should be pointed out that to keep the arithmetic in the real domain for real matrices, each QZ iteration is a double step. Thus a fairer comparison might be to compare one QZ iteration to two LZ iterations and to keep in mind that even for real matrices LZ uses complex arithmetic. For complex matrices a single shift version of QZ is probably preferable to a double shift version of QZ. A single shift QZ iteration would require $6n^2$ multiplications and $6n^2$ additions and 2n square roots and an extra $3n^2$ multiplications and $3n^2$ additions if eigenvectors are requested. These statistics seem to indicate that the LZ method is the more efficient than the QZ method for complex matrices.

It is also interesting to compare each iteration of the LZ algorithm with each iteration of the standard LR algorithm as given in [17]. The standard LR requires n^2 additions and n^2 multiplications per iteration, and n^2 more multiplications and n^2 more additions if eigenvectors are requested. Thus the basic LZ algorithm does twice as much work per iteration as the LR method, but only 3/2 times as much work when the accumulation of matrices to obtain eigenvectors is considered.

The operation counts given above can be summarized as follows:

OPERATION COUNTS PER ITERATION

	Without Eigenvectors			With Eigenvectors		
	+	×	Square roots	+	×	Square roots
*LZ	$2n^2$	$2n^2$	0	$3n^2$	$3n^2$	0
Double QZ	$13n^2$	$13n^2$	$3n$	$21n^2$	$21n^2$	$3n$
*Single QZ	$6n^2$	$6n^2$	$2n$	$9n^2$	$9n^2$	$2n$
*LR	n^2	n^2	0	$2n^2$	$2n^2$	0

* Always uses complex arithmetic.

The operation counts reported for the double QZ algorithm are those given in [10]. If the left eigenvectors of the problem with transposed A and B are computed, as opposed to the right eigenvectors of the original problem, then the left hand transformations must be accumulated and only $18n^2$ additions and $18n^2$ multiplications are required per iteration.

III. PROOF OF THE EQUIVALENCE OF THE IMPLICIT AND EXPLICIT SCHEMES

In this section we will prove that the explicit and implicit schemes, described in the previous section, generate and use the same transformations.

Theorem. Let $\underline{L}_j$ and $\underline{M}_j$ represent the transformation in the implicit method and L_j and M_j represent the transformations in the explicit method. If the next iterate is A' and $a'_{j,j-1}$ is nonzero for $j < i$, then for $j < i$, $\underline{L}_j = L_j$ and $\underline{M}_j = M_j$.

Proof. The proof is by induction on j. By construction $\underline{L}_1$ is equal to L_1 and $\underline{M}_1$ is equal to M_1. We assume that $\underline{L}_k = L_k$ and $\underline{M}_k = M_k$ for $k < j$, and let

$$A^* = L_{j-1} \cdot \cdot \cdot L_1 A M_1 \cdot \cdot \cdot M_{j-1}$$

$$B^* = L_{j-1} \cdot \cdot \cdot L_1 B M_1 \cdot \cdot \cdot M_{j-1}$$

$$T^* = L_{j-1} \cdot \cdot \cdot L_1 T M_1 \cdot \cdot \cdot M_{j-2}$$

$$A^{**} = L_j A^*$$

	A*	T*	B*
	X X X X X	X X X X X	X X X X X
	X X X X X	X X X X X	0 X X X X
j^{th} row →	0 X X X X	0 0 X X X	0 0 X X X
	0 0 X X X	0 0 0 X X	0 0 0 X X
	0 0 0 X X	0 0 0 X X	0 0 0 0 X

j^{th} column

Because $T = A - \lambda B$, we must have

$$A^* = T^*M_{j-1} + \lambda B^*$$

which implies that

$$A^{**} = L_jA^* = L_jT^*M_{j-1} + \lambda L_jB^*.$$

We know that $L_jT^*M_{j-1}$ is in $\mathcal{H}$. Since L_jB^* is also in $\mathcal{H}$, A^{**} must also be in $\mathcal{H}$. But in the implicit method $\underline{L}_j$ is the transformation from $\mathcal{L}_j$ which returns A^* to upper Hessenberg form and if either $a^*_{j+1,j-1}$ or $a^*_{j,j-1}$ is nonzero, there is only one element belonging to $\mathcal{L}_j$ which can accomplish this. Since transformations to A occuring after L_j do not affect the $j-1^{st}$ column of A, the element $a'_{j,j-1}$ is nonzero only if $a^{**}_{j,j-1}$ is nonzero. Since $a^{**}_{j,j-1}$ is zero only if both $a^*_{j,j-1}$ and $a^*_{j+1,j-1}$ are zero, the hypothesis to our theorem assures us that there is only one transformation from $\mathcal{L}_j$ which could return A^* to an element of $\mathcal{H}$. Since both L_j and $\underline{L}_j$ belong to $\mathcal{L}_j$, we know they must be identical. By construction M_j and $\underline{M}_j$ must also be identical and therefore we have proved our theorem by induction. ∎

If $a'_{j,j-1}$ is zero then we have no assurance that row and column transformations subsequent to M_{j-1} in the explicit and implicit algorithms are identical, but this is of little consequence. In fact, the best policy in both algorithms is that as soon as a permanent zero is detected on the subdiagonal of A', then the iteration should be discontinued and work begun on a problem of lower dimension. If in both methods, this policy were adopted, then the algorithms would be equivalent up to roundoff error.

IV. FINDING THE EIGENVECTORS

After A and B have been reduced to triangular form, the eigenvectors can be easily determined. Let α_j, and β_j denote the diagonal elements of the triangularized A and B and let $\underset{\sim}{y}_j$ denote the corresponding eigenvector, that is

$$(\beta_j A - \alpha_j B)\, \underset{\sim}{y}_j = 0 .$$

The components y_{ij} of $\underset{\sim}{y}_j$ can be obtained by solving this triangular system as follows:

$$y_{ij} = 0 \quad \text{for} \quad i < j$$

$$y_{ij} = 1 \text{ for } \quad i = j$$

$$y_{ij} = \frac{1}{(\beta_j \alpha_i - \alpha_j \beta_i)} \sum_{k=i+1}^{j} (\beta_j a_{ik} - \alpha_j b_{ik})\, y_{kj}$$

$$\text{for} \quad i = j-1, j-2, \ldots 1.$$

The jth eigenvector of the original system can be found by multiplying $\underset{\sim}{y}_j$ by M.

If the denominator in the above formula is zero, then it is replaced by macheps $*(\|A\|_\infty + \|B\|_\infty)$. The denominator is zero when the ith and j^{th} eigenvalues are equal. If the numerator is also zero, then linearly independent solutions will be produced. However, if the numerator is not zero, then $\underset{\sim}{y}_j$ will have large components and after normalization $\underset{\sim}{y}_i$ and $\underset{\sim}{y}_j$ will be nearly linearly dependent. This occurs when the eigenvalue does not have a full set of eigenvectors. See Peters and Wilkinson[16] for further discussion.

CHAPTER TWO

CONVERGENCE RESULTS

In this chapter we will prove several convergence theorems, each of which restricts the problem in some way. A theorem might specify some property of the eigenvalues themselves or characterize the matrices L and M. Usually both types of restrictions are given. Many of the theorems refer to a nonshifting or constant shifting version of the LZ algorithm. We have found that in most instances when a shift policy has not worked, the method has been using the same shift for each iteration. The chapter ends with a partial listing of 3 x 3 examples for which the current algorithm, which uses a solution of the lower 2 x 2 problem of $A\underset{\sim}{x} = \lambda B\underset{\sim}{x}$ as a shift, will not converge without the use of an appropriate ad-hoc shift.

We will use Parlett's [11] terminology and say that a matrix is an Unreduced Hessenberg matrix (UHM) if it is an upper Hessenberg matrix and none of its subdiagonal elements is zero. To simplify our proofs we will assume we are working with the algorithm in its explicit form. The matrices will be n x n, and unless stated otherwise, we will assume that A is a UHM and B is triangular. For uniformity we will assume that the kth iteration in the algorithm is given by

1) Find a shift λ_k.
2) Form $T_k = A_k - \lambda_k B_k$.
3) Find $\bar{L}_k$ such that $\bar{L}_k T_k$ is triangular.
4) Find $\bar{M}_k$ such that $\bar{L}_k B_k \bar{M}_k$ is upper triangular.
5) Set $B_{k+1} = \bar{L}_k B_k \bar{M}_k$, $A_{k+1} = \bar{L}_k T_k \bar{M}_k + \lambda_k B_{k+1}$.

$\overline{L}_k$ is a product of matrices $L_{k,n-1}L_{k,n-2}\cdots\cdots L_{k,1}$ and $\overline{M}_k$ is a product of matrices $M_{k,1}\ M_{k,2}\cdots\cdots M_{k,n-1}$ where each $L_{k,i}$ is an element of $\mathcal{L}_i$, and each $M_{k,i}$ is in $\mathcal{M}_i$ as described in the notation section of the introduction. In this chapter the multiplier in $L_{k,i}$ will be denoted by $\eta_{k,i}$ and the multiplier in $M_{k,i}$ will be denoted by $\mu_{k,i}$.

In many of our theorems we will drop the iteration subscript and designate the matrices A_k, B_k, etc. by A, B, and the matrices A_{k+1}, B_{k+1}, etc. by A', B'. The matrices $L_{k,i}$ and $M_{k,i}$ will be denoted by L_i and M_i respectively, and their corresponding multipliers by η_i and μ_i.

I. DEFINITION OF CONVERGENCE

In general the algorithm will be said to be "convergent" if, as k approaches infinity, one of the elements on the subdiagonal of A_k approaches zero or if for some finite k, one of the elements on the subdiagonal of A_k is zero. Because of the interrelationships that exist between A_k, T_k, L_k, and M_k, we will also regard the algorithm as convergent if one of the following conditions is satisfied:

1) As k increases, one of the elements on the subdiagonal of T_k approaches zero; or for some finite k, one of the elements on the subdiagonal of T_k equals 0.

2) For a fixed i, as k increases, $L_{k,i}$ approaches the identity matrix.

3) For a fixed i, as k increases, $M_{k,i}$ approaches the identity matrix.

The reason for the first criterion is that in the explicit algorithm the subdiagonal elements of A_k and T_k are identical. The reason for the second criterion is that if $t^{(k)}_{i+1,i}$ is zero, then $L_{k,i}$ will be the identity matrix, and the reason for the third condition is that if $M_{k,i}$ is I, then $a^{(k+1)}_{i+1,i}$ will be zero.

It should be emphasized that when we say LZ converges, we mean that the problem can be divided into two subproblems of lower dimension. We do not necessarily mean the algorithm can find all the eigenvalues.

II. SINGULAR MATRICES

We begin by proving a theorem which substantiates our claim that LZ works even when B is singular.

Theorem 1: If B is singular, the LZ algorithm converges in at most n iterations if exact arithmetic is used.

Proof: If B is singular, one of its diagonal elements must be zero. If b_{11} is zero, then applying L_1 to B will not change the 0 in the (2,1) position of B. Thus M_1 will be the identity matrix I and the LZ algorithm will converge immediately.

Now let us assume that $b_{ii} = 0$ and $i > 1$ and let us look at the 2 by 2 matrix formed by the $i-1^{st}$ and i^{th} rows and columns of B. It will look like

$$\begin{pmatrix} a & b \\ 0 & 0 \end{pmatrix}.$$

If L_{i-1} involves pivoting this matrix will become

$$\begin{pmatrix} 0 & 0 \\ a & b \end{pmatrix}$$

and, independent of the form of M_{i-1}, the 0 will remain in the (i-1,i-1) position. Future transformations of B during this iteration will not affect this 0.

If L_{i-1} does not involve pivoting, then we get

$$\begin{pmatrix} a & b \\ \eta_{i-1}a & \eta_{i-1}b \end{pmatrix}.$$

If a is zero, M_{i-1} is the identity matrix and $b_{i-1,i-1}$ is 0. If b is zero, then M_{i-1} permutes the ith and $i-1^{st}$ columns, and $b_{i-1,i-1}$ is also 0. If b is nonzero and a is nonzero, then $b_{i-1,i-1}$ is either $a-(\eta_{i-1}a/(\eta_{i-1}b))b$ or $b-(\eta_{i-1}b/\eta_{i-1}a))b$. In either case, it is zero if the arithmetic is exact. Again future transformations on B during this iteration can not affect the zero in the $(i-1,i-1)$ position.

We see that with each iteration, a zero on the diagonal of B moves up one row. Within $n-1$ iterations it must reach the $(1,1)$ position, at which point the algorithm must converge in one iteration. ∎

The following lemmas consider the case in which B is nearly singular. The quantity ϵ is assumed to be a small number relative to the norm of B.

<u>Lemma 1</u>: If $b_{i,i} = \epsilon$ for $i > 1$ and L_{i-1} involves pivoting then $\| b'_{i-1,i-1} \| \leq \|\epsilon\|$

Proof: If B^* represents the matrix

$$L_{i-1}L_{i-2}\cdots L_1 B M_1 \cdots M_{i-2}$$

then $b^*_{i-1,i} = \epsilon$ and $b^*_{i-1,i-1} = 0$. Thus $b'_{i-1,i-1}$ will either be ϵ or $\mu_{i-1}\epsilon$, and since $\| \mu_{i-1} \| \leq 1$, $\| b'_{i-1,i-1} \|$ will be $\leq \|\epsilon\|$.

<u>Lemma 2</u>: If $b_{i,i} = \epsilon$ for $i > 1$ and L_{i-1} does not involve pivoting, then

$$\| b'_{i-1,i-1} \| \leq \left\| \frac{\epsilon}{\eta_{i-1}} \right\| .$$

Proof: If B^* is the matrix

$$L_{i-1}L_{i-2}\cdot\cdot\cdot L_1BM_1\cdot\cdot\cdot M_{i-2}$$

then we may write

$$b^*_{i-1,i-1} = a \qquad b^*_{i-1,i} = b$$

$$b^*_{i,i-1} = \eta_{i-1}a \qquad b^*_{i,i} = \epsilon+\eta_{i-1}b.$$

If M_{i-1} does not involve pivoting, then

$$b'_{i-1,i-1} = a - \frac{\eta_{i-1}a\ b}{\epsilon + \eta_{i-1}b}$$

$$= \frac{a\epsilon}{\epsilon + \eta_{i-1}b} = \frac{\epsilon}{\eta_{i-1}} \cdot \frac{\eta_{i-1}a}{(\epsilon + \eta_{i-1}b)}$$

and hence

$$\|b'_{i-1,i-1}\| \leq \|\frac{\epsilon}{\eta_{i-1}}\|.$$

If M_{i-1} does involve pivoting then

$$b'_{i-1,i-1} = b - (\epsilon+\eta_{i-1}b)a/(\eta_{i-1}a)$$

$$= -\frac{\epsilon}{\eta_{i-1}} \quad . \blacksquare$$

The previous two lemmas indicate that small elements on the diagonal of B creep up the diagonal of B. The next lemma gives us an idea of what occurs when the small element reaches the top of B.

Lemma 3: If $b_{1,1}^{(k)} = \epsilon$ and $L_{k+i,1}$ and $M_{k+i,1}$ do not involve pivoting for $i \leq j$, then there are constants c_j and d_j such that

$$b_{1,1}^{(k+j)} = c_j\epsilon \quad \text{and} \quad a_{2,1}^{(k+j)} = d_j\epsilon^j$$

for $j \geq 0$.

Proof: Our proof is by induction on j. For j = 1, the hypothesis to our lemma implies that

$$\mu_{k,1} = -\frac{\epsilon\,\eta_{k,1}}{b_{2,2}^{(k)} + \eta_{k,1}b_{1,2}^{(k)}} \equiv c\,\epsilon$$

which means that

$$b_{1,1}^{(k+1)} = \epsilon\,(1 + cb_{1,2}^{(k)}) \equiv c_1\epsilon$$

and if $L_{k,2}$ does not involve pivoting

$$a_{2,1}^{(k+1)} = c\epsilon\,(t_{2,2}^{(k)} + \eta_{k,1}t_{1,2}^{(k)}) \equiv d_1\epsilon$$

and if $L_{k,2}$ does involve pivoting

$$a_{2,1}^{(k+1)} = c\epsilon t_{2,2}^{(k)} - d_1\epsilon.$$

If we assume our lemma is true for j, then

$$\eta_{k+j,1} = -d_j\epsilon^j / t_{11}^{(k+j)}$$

which means that

$$\mu_{k+j,1} = \frac{d_j c_j \epsilon^{j+1} / t_{1,1}^{(k+j)}}{b_{2,2}^{(k+j)} + \eta_{k+j,1}b_{1,2}^{(k+j)}}$$

$$\equiv c\epsilon^{j+1}$$

so that

$$b_{1,1}^{(k+j+1)} = \epsilon(c_j + cb_{1,2}^{(k+j)}) \equiv c_{j+1}\epsilon$$

If $L_{k+j,2}$ does not use pivoting

$$a_{2,1}^{(k+j+1)} = c\epsilon^{j+1}(t_{2,2}^{(k+j)} + \eta_{k+j,1}t_{1,2}^{(k+j)})$$

$$\equiv d_{j+1}\epsilon^{j+1}.$$

If $L_{k,2}$ uses pivoting

$$a_{2,1}^{(k+j+1)} = c\epsilon^{j+1}\, t_{j,2}^{(k+j)} \equiv d_{j+1}\epsilon^{j+1}. \blacksquare$$

Our lemma did not state the size of the elements c_j and d_j and there is no guarantee that they will be small.

Our next theorem is a counterpart of Theorem 1. We note that B might be singular.

<u>Theorem 2</u>: If T is singular, i.e. λ is an eigenvalue of the problem $A\underset{\sim}{x} = \lambda B\underset{\sim}{x}$, then the LZ algorithm converges in one step in exact arithmetic.

Proof: The first n-1 columns of T must be linearly independent or else some subdiagonal element of T would be zero thus implying T is not a UHM. The algorithm constructs a nonsingular matrix L such that LT = R, an upper triangular matrix. Since the first n-1 columns of T are linearly independent, the first n-1 columns of R must also be. Similarly, since T is singular, R must also be. This means that the last column of R may be written as a linear combination of the first n-1 columns, and because the last component of each of the first n-1 columns of R is zero, the last component of its nth column must also be zero. Hence the last row of R is $\underset{\sim}{\theta}^T$, the null vector.

Now in the next step of the algorithm we construct M so that $B' = LBM$ is upper triangular and set $A' = T' + kB'$ where $T' = LTM$. Since the last row of R is $\underset{\sim}{\theta}^T$, the last row of T' must also be $\underset{\sim}{\theta}^T$, and hence $a'_{n,n-1}$ must be zero. ∎

Our theorems and lemmas indicate that in the future we can safely ignore problems where either A or B is singular. However, we would like to include singular cases in the next few theorems wherever possible, because many of these theorems not only guarantee convergence, they also give some hints about rates of convergence.

III. GLOBAL CONVERGENCE

In this section, we shall prove several global convergence theorems. Most of our results refer to a constant shifting algorithm, although in practice our shifts are usually different for each iteration. However, we have often found that when a shift policy has not worked, the same shift is being used for each iteration. Thus, our theorems do have practical significance although they do not refer to any actual implementation of LZ.

We begin by proving four similar theorems. They all consider the sequences of matrices $\{A_k\}$, $\{B_k\}$, $\{\bar{L}_k\}$, $\{\bar{M}_k\}$, $\{T_k\}$ and $\{S_k\}$

where $T_k = A_k - \lambda_k B_k$ for some scalar λ_k

$$A_{k+1} = \bar{L}_k A_k \bar{M}_k \qquad A_1 = A$$

$$B_{k+1} = \bar{L}_k B_k \bar{M}_k \qquad B_1 = B$$

$$S_k = \bar{L}_k T_k$$

and $\bar{L}_k$ and $\bar{M}_k$ are nonsingular for all k and S_k and B_k are upper triangular for all k.

<u>Theorem</u>: If B_1 is nonsingular,

$$V_k = \bar{L}_1^{-1} \bar{L}_2^{-1} \cdots\cdots\cdots \bar{L}_k^{-1},$$

and

$$U_k = S_k B_k^{-1} S_{k-1} B_{k-1}^{-1} \cdots\cdots\cdots S_1 B_1^{-1}$$

then

$$V_k U_k = (A - \lambda_k B) B^{-1} (A - \lambda_{k-1} B) B^{-1} \cdots (A - \lambda_1 B) B^{-1} .$$

Proof: Since

$$T_k = \bar{L}_{k-1}(A_{k-1} - \lambda_k B_{k-1})\bar{M}_{k-1}$$

and

$$B_k^{-1} = \bar{M}_{k-1}^{-1} B_{k-1}^{-1} \bar{L}_{k-1}^{-1}$$

we have

$$T_k B_k^{-1} = \bar{L}_{k-1}(A_{k-1} - \lambda_k B_{k-1}) B_{k-1}^{-1} \bar{L}_{k-1}^{-1}$$

$$= V_{k-1}^{-1}(A_1 - \lambda_k B_1) B_1^{-1} V_{k-1}$$

which implies

$$V_{k-1} T_k B_k^{-1} = (A - \lambda_k B) B^{-1} V_{k-1} .$$

Now $V_k U_k = V_{k-1} \bar{L}_k^{-1} S_k B_k^{-1} U_{k-1}$

$$= V_{k-1} \bar{L}_k^{-1} \bar{L}_k T_k B_k^{-1} U_{k-1}$$

$$= V_{k-1} T_k B_k^{-1} U_{k-1}$$

$$= (A - \lambda_k B) B_1^{-1} V_{k-1} U_{k-1}$$

$$= (A - \lambda_k B) B^{-1} (A - \lambda_{k-1} B) B^{-1} \ldots (A - \lambda_2 B) B^{-1} (A - \lambda_1 B) B^{-1} . \blacksquare$$

<u>Corollary 3.1</u>. If the conditions of Theorem 3 are satisfied and $\lambda_k = \rho$ for all k, then $V_k \underset{\sim}{e}_1$ is in the direction of $((A - \rho B)B^{-1})^k \underset{\sim}{e}_1$.

Proof. By Theorem 3. $V_k U_k = ((A - \rho B)B^{-1})^k$. Since U_k is upper triangular, $U_k \underset{\sim}{e}_1 = u_{1,1}^{(k)} \underset{\sim}{e}_1$, and hence

$$V_k \; u_{1,1}^{(k)} \; \underset{\sim}{e}_1 = ((A - \rho B)B^{-1})^k \underset{\sim}{e}_1 . \quad \blacksquare$$

<u>Theorem 4.</u> If $A - \lambda_i B$ is nonsingular for all i, then

$$U_k^{-1} V_k^{-1} = B\,(A - \lambda_1 B)^{-1} B\,(A - \lambda_2 B)^{-1} \cdot \cdot \cdot \cdot B\,(A - \lambda_k B)^{-1}.$$

The proof of the above theorem parallels that of the previous theorem.

<u>Corollary 4.1.</u> If the conditions of Theorem 4 are satisfied and $\lambda_k = \rho$ for all k, then $V_k^{-T} \underset{\sim}{e}_n$ is in the direction of $((A^T - \rho B^T)^{-1} B^T)^k \underset{\sim}{e}_n$.

Proof. By theorem 4

$$U_k^{-1} V_k^{-1} = [B(A - \rho B)^{-1}]^k$$

which means that

$$V_k^{-T} U_k^{-T} = ((A^T - \rho B^T)^{-1} B^T)^k .$$

Since U_k is the product of upper triangular matrices, the matrix $P_k \equiv U_k^{-T}$ must be lower triangular which implies that $P_k \underset{\sim}{e}_n = p_{n,n}^{(k)} \underset{\sim}{e}_n$ and hence

$$V_k^{-T} \; p_{n,n}^{(k)} \; \underset{\sim}{e}_n = ((A^T - \rho B^T)^{-1} B^T)^k \underset{\sim}{e}_n . \quad \blacksquare$$

If the matrices described above represented the matrices in a version of the LZ algorithm, which did not allow row interchanges, then for all k, the matrix V_k would be unit lower triangular.

Theorem 5. If B is nonsingular,

$$\bar{U}_k = B_{k+1}^{-1} S_k B_k^{-1} \cdots\cdots\cdots\cdots B_2^{-1} S_1$$

and

$$W_k = \bar{M}_1 \bar{M}_2 \cdots\cdots \bar{M}_k$$

then

$$W_k U_k = B_1^{-1}(A_1 - \lambda_k B_1) B_1^{-1}(A_1 - \lambda_{k-1} B_1) \cdot \cdot \cdot \cdot B_1^{-1}(A_1 - \lambda_1 B_1).$$

Proof. Since

$$T_k = \bar{L}_{k-1}(A_{k-1} - \lambda_k B_{k-1})\bar{M}_{k-1}$$

and

$$B_k^{-1} = \bar{M}_{k-1}^{-1} B_{k-1}^{-1} \bar{L}_{k-1}^{-1},$$

we have

$$B_k^{-1} T_k = \bar{M}_{k-1}^{-1} B_{k-1}^{-1}(A_{k-1} - \lambda_k B_{k-1})\bar{M}_{k-1}$$

$$= W_{k-1}^{-1} B_1^{-1}(A_1 - \lambda_k B_1) W_{k-1}$$

which implies

$$W_{k-1} B_k^{-1} T_k = B_1^{-1}(A \quad - \lambda_k B \quad) W_{k-1} \cdot$$

Now

$$W_k \bar{U}_k = W_{k-1} \bar{M}_k B_{k+1}^{-1} S_k \bar{U}_{k-1}$$

$$= W_{k-1} \bar{M}_k \bar{M}_k^{-1} B_k^{-1} L_k^{-1} S_k \bar{U}_{k-1}$$

$$= W_{k-1} B_k^{-1} T_k \bar{U}_{k-1}$$

$$= B^{-1}(A \quad - \lambda_k B \quad) W_{k-1} \bar{U}_{k-1}$$

$$= B^{-1}(A \quad - \lambda_k B \quad) B^{-1}(A \quad - \lambda_{k-1} B \quad) \cdot \cdot \cdot \cdot B^{-1}(A \quad - \lambda \quad B \quad).$$

Corollary 5.1. If the hypothesis of Theorem 5 is satisfied and $\lambda_k = \rho$ for all k, then $W_k \underset{\sim}{e}_1$ is in the direction of $(B^{-1}(A - \rho B))^k \underset{\sim}{e}_1$.

Proof. By Theorem 5

$$W_k \bar{U}_k = (B^{-1}(A - \rho B))^k.$$

Since $\bar{U}_k$ is a product of upper triangular matrices, $\bar{U}_k \underset{\sim}{e}_1 = \bar{u}_{1,1}^{(k)} \underset{\sim}{e}_1$ which means that

$$W_k \; \bar{u}_{1,1}^{(k)} \underset{\sim}{e}_1 = (B^{-1}(A - \rho B))^k \underset{\sim}{e}_1. \quad \blacksquare$$

Theorem 6. If $A - \lambda_k B$ is nonsingular for all i, then

$$\bar{U}_k^{-1} W_k^{-1} = (A - \lambda_1 B)^{-1} B (A - \lambda_2 B)^{-1} B \ldots (A - \lambda_k B)^{-1} B .$$

The proof of this theorem parallels that of the previous theorem. Note that this theorem does not assume that B is nonsingular.

Corollary 6.1. If the hypothesis of Theorem 6 is satisfied and $\lambda_k = \rho$ for all k then $W_k^{-T} \underset{\sim}{e}_n$ is in the direction of $(B^T(A^T - \rho B^T)^{-1})^k \underset{\sim}{e}_n$.

Proof. By Theorem 6

$$\bar{U}_k^{-1} W_k^{-1} = ((A - \rho B)^{-1} B)^k$$

which means that

$$W_k^{-T} \bar{U}_k^{-T} = (B^T (A - \rho B)^{-T})^k .$$

Since $\bar{U}_k$ is a product of upper triangular matrices, $Q_k \equiv \bar{U}_k^{-T}$ is lower triangular and $Q_k \underset{\sim}{e}_n$ equals $q_{n,n}^{(k)} \underset{\sim}{e}_n$. This in turn implies that

$$W_k^{-T} q_{n,n}^{(k)} \underset{\sim}{e}_n = (B^T (A^T - \rho B^T)^{-1})^k \underset{\sim}{e}_n .$$

If we again relate the above theorems to the LZ algorithm, then W_k would be unit lower triangular if column interchanges were not permitted.

If D is a diagonal matrix with diagonal elements $d_1, d_2, \ldots d_n$ then the Moore-Penrose pseudo inverse of D, is the diagonal matrix, denoted by D^+, with elements $z_1, z_2, \ldots z_n$ where

$$z_i = 1/d_i \quad \text{if } d_i \text{ is nonzero}$$

and

$$z_i = 0 \quad \text{if } d_i \text{ is zero.}$$

We will denote by Φ the projection matrix $D^+ D$.

$$\Phi_{ij} = \begin{cases} 1 & \text{for } i = j \text{ and } d_i \text{ is nonzero.} \\ 0 & \text{elsewhere} \end{cases}$$

The next main theorem is a modification of one that appears in Wilkinson [19] and Parlett [11] among other places. Our version of the theorem extends their results to cover singular matrices. If a matrix X is said to have an LU decomposition, then there exists an upper triangular matrix U and a lower triangular matrix L such that $X = LU$. This decomposition is unique.

<u>Theorem 7</u>. If F is a matrix with eigenvalues of distinct modulus satisfying

$$|d_1| > |d_2| > \cdot \cdot \cdot \cdot > |d_n| \geq 0,$$

and if F can be written as XDY where $Y = X^{-1}$, D is the diagonal matrix of eigenvalues and Y has an LU decomposition and X has an LU decomposition $L_x U_x$, then the lower triangular factor of the LU decomposition of F^k goes to L_Φ as $k \to \infty$.

To facilitate the proof of Theorem 7, we first present a few lemmas.

<u>Lemma 4</u>. If L is a lower triangular matrix and D is a diagonal matrix whose elements satisfy

$$|d_1| > |d_2| > \cdot \cdot \cdot > |d_m| > |d_{m+1}| = \cdot \cdot \cdot = |d_n|$$

then $$DL = DLD^{+}D.$$

Proof. If $S = DL$, then $s_{j,j} = 0$ for $j > i$ or $i > m$ and $s_{i,j} = d_i \ell_{i,j}$ elsewhere.

Now the last n-m columns of S are zero because for $j > m$,

$$s_{i,j} = 0 \quad \text{for} \quad i > m$$

and

$$s_{i,j} = 0 \quad \text{for} \quad i < j, \text{ i.e. for } i \leq m.$$

Multiplying S by $D^{+}D$ zeroes the last n-m columns of S and leaves the first m columns untouched. Since the last n-m columns of S are already zero we have $S = SD^{+}D$ or

$$DL = DLD^{+}D. \blacksquare$$

<u>Lemma 5</u>. Let L be a unit lower triangular matrix, and let D be the matrix of Lemma 4, and $G_k = D^k L D^{+k}$;

then $$G_k = \Phi + E_k \quad \text{where} \quad E_k \to 0 \quad \text{as} \quad k \to \infty.$$

Proof. If $g_{i,j}^{(k)}$ is the (i,j) element of G_k, then

$$g_{ij}^{(k)} = \begin{cases} 0 \text{ for } i < j \text{ or } j > m \\ \ell_{i,j} \, (d_i/d_j)^k \text{ elsewhere} \end{cases}.$$

The fact that $|d_i/d_j| < 1$ for $i > j$ and $j \leq m$ and $d_j/d_i = 1$ for $i \leq m$, together with the fact that L is unit lower triangular implies that $g_{jj}^{(k)} = 1$ for $j \leq m$ and $g_{ij}^{(k)} \to 0$ for $j < i$ and $j \leq m$ as k approaches infinity. Thus $G_k = \Phi + E_k$ where $E_k \to 0$ as k approaches infinity. ∎

<u>Lemma 6</u>. If U is an upper triangular matrix and D is the diagonal matrix of Lemma 4, then

$$U\Phi D = \Phi U D .$$

Proof: If $G = U\Phi$, then

$$g_{i,j} = 0 \quad \text{for } j > m \text{ or } i > j$$

and

$$g_{i,j} = u_{i,j} \quad \text{elsewhere.}$$

if $H = GD$, then

$$h_{i,j} = 0 \quad \text{for } j > m \text{ or } i > j$$

and

$$h_{i,j} = u_{i,j} d_j \quad \text{elsewhere.}$$

The formulae just given for $h_{i,j}$ are exactly those which would be given for $e_{i,j}$ where $E = UD$. Thus $H = E$. Moreover, the last n-m rows of E are zero so that $\Phi E = E$. Thus

$$\Phi E = H \quad \text{or} \quad \Phi UD = U\Phi D. \ ∎$$

We can now give the proof of theorem 7.

Proof of theorem 7: Assume the eigenvalues of the matrix F are of distinct moduli and assume F can be written as $F = XDX^{-1}$ where D is the diagonal matrix of eigenvalues and both X and X^{-1} have LU decompositions. Let $L_x U_x$ be the LU decomposition of X and let $L_y U_y$ be the LU decomposition of X^{-1}. It can be easily shown

that
$$F^k = XD^kX^{-1}$$
$$= XD^kL_yU_y$$
$$= XD^kL_yD^{+k}D^kU_y \quad \text{by Lemma 4.}$$

If $G_k = D^kL_yD^{+k}$, then by Lemma 5 $G_k = \Phi + E_k$ where E_k goes to 0 as k goes to infinity.

Thus,
$$F^k = X(\Phi + E_k)D^kU_y$$
$$= L_xU_x(\Phi + E_k)D^kU_y$$
$$= L_x(\Phi + U_xE_kU_x^{-1})U_xD^kU_y \quad \text{by Lemma 6.}$$

Since $U_xE_kU_x^{-1}$ goes to 0 as k goes to infinity, $\Phi + U_xE_kU_x^{-1}$ goes to Φ as $k \to \infty$. Because $\Phi U_xD^kU_y$ is upper triangular, the lower triangular factor of F^k approaches $L_x\Phi$ as k becomes large. ∎

<u>Theorem 8</u> : If

(1) B is nonsingular

(2) LZ is used with a constant shift ρ.

(3) The quantities $\lambda_i - \rho$ for $i = 1,2,\ldots n$ have distinct moduli

(4) Either no row pivoting is required and there exists a matrix X such that $(A - \rho B)B^{-1} = XDX^{-1}$ where D is diagonal and both X and X^{-1} have LU decompositions or no column pivoting is required and there exists a matrix X such that $B^{-1}(A - \rho B) = XDX^{-1}$ where D is diagonal and both X and X^{-1} have LU decompositions,

then LZ converges.

Proof: We let $F = (A - \rho B)B^{-1}$ and apply Theorem 7 and find that as k increases, the lower triangular factor of the LU decomposition of F^k approaches L_{X^Φ}. By Theorem 3 this lower triangular factor is given by V_k. If $V_k \to L_{X^\Phi}$ and $V_{k+1} \to L_{X^\Phi}$ and $V_{k+1} = V_k \bar{L}_k^{-1}$, then $\bar{L}_k^{-1}$ must be approaching the identity matrix which means that LZ is convergent.

If $F = B^{-1}(A - \rho B)$ then by invoking Theorem 7 and Theorem 5 we see that $\bar{M}_k$ must be approaching the identity matrix as $k \to \infty$ which means that LZ is convergent. ∎

The condition that both X and X^{-1} have LU decompositions is partially satisfied since both $(A - \rho B)B^{-1}$ and $B^{-1}(A - \rho B)$ are unreduced Hessenberg matrices. Parlett [14] has proved that if F is a UHM, then there exists a matrix X such that XJX^{-1} is the Jordan canonical form of F and X^{-1} has an LU decomposition.

The condition on the distinctness of the moduli of the eigenvalues can be relaxed somewhat. Wilkinson's [19] treatment of multiple eigenvalues for the LR algorithm can be applied directly to the LZ algorithm.

IV. ITERATION AND LZ.

Theorem 8 indicates that with a constant shift the LZ algorithm converges for problems with shifted eigenvalues of distinct moduli if row pivoting is not required or if column pivoting is not required. Some of these restrictions can be weakened by investigating the relationships between LZ and various iterative procedures. This approach has been taken by Poole [13], Parlett and Kahan [12], and Buurema [1] in studying the QR algorithm.

Let us consider the LZ algorithm with constant shift ρ for the problem having eigenvalues λ_i and let us denote $\lambda_i - \rho$ by d_i and assume that the $\{\lambda_i\}$'s are ordered so that

$$|d_1| \geq |d_2| \geq \cdot \cdot \cdot \cdot \geq |d_n| \ .$$

Theorem 9. If any of the following 4 conditions is satisfied then the algorithm converges.

1) $|d_1| \neq |d_2|$, $\underset{\sim}{e}_1$ is not orthogonal to the right eigenvector of $(A - \rho B)B^{-1}$ associated with d_1 and row pivoting is never done to zero $t_{2,1}^{(k)}$ for all k.

2) $|d_n| \neq |d_{n-1}|$, $\underset{\sim}{e}_n$ is not orthogonal to the left eigenvector of $B(A - \rho B)^{-1}$, associated with $1/d_n$ and row pivoting is never done to zero $t_{n,n-1}^{(k)}$ for all k.

3) $|d_1| \neq |d_2|$, $\underset{\sim}{e}_1$ is not orthogonal to the right eigenvector of $B^{-1}(A - \rho B)$ associated with d_1 and column pivoting is never done in zeroing $b_{2,1}^{(k)}$ for all k.

4) $|d_n| \neq |d_{n-1}|$, $\underset{\sim}{e}_n$ is not orthogonal to the left eigenvector of $(A - \rho B)^{-1}B$ associated with $1/d_n$ and column pivoting is never done to zero $b_{n,n-1}^{(k)}$ for all k.

Proof. The proof of (1) entails looking at the power method for solving $(A - \rho B)\underset{\sim}{x} = d\, B\underset{\sim}{x}$ given by

1) Set $\underset{\sim}{x}_0$ to $\underset{\sim}{e}_1$ and k to 0.

2) Find $\underset{\sim}{y}$ such that $B\underset{\sim}{y} = \underset{\sim}{x}_k$.

3) Set $\underset{\sim}{z}$ to $(A - \rho B)\underset{\sim}{y}$.

4) Set $\underset{\sim}{x}_{k+1} = \underset{\sim}{z}/\|\underset{\sim}{z}\|$.

If $\|\underset{\sim}{x}_{k+1} - \underset{\sim}{x}_k\|$ is small then stop;

otherwise, set k to $k+1$ and return to 2.

If $|d_1|>|d_2|$ and $\underset{\sim}{e}_1$ is not orthogonal to the left eigenvector of $(A - \rho B)B^{-1}$ corresponding to d_1, then the power method converges and $\underset{\sim}{x}_k$ will be that eigenvector. We note that $\underset{\sim}{x}_k$ is in the direction of $((A - \rho B)B^{-1})^k \underset{\sim}{e}_1$, which, according to Corollary 3.1, is also the direction of $V_k \underset{\sim}{e}_1$. Thus if the power method converges, it should be clear that

$$V_{k+1}\underset{\sim}{e}_1 = c_k V_k \underset{\sim}{e}_1 + \underset{\sim}{f}_k \qquad (2 - 1)$$

where c_k is some scalar and $\underset{\sim}{f}_k \to \underset{\sim}{\theta}$ as $k \to \infty$.

Since $V_{k+1} = V_k \bar{L}_k^{-1}$

and $\bar{L}_k^{-1} = L_{k,1}^{-1} L_{k,2}^{-1} \cdots\cdots\cdots L_{k,n-1}^{-1}$ where $L_{k,i}$ is in $\mathcal{S}_i$

which means that $\bar{L}_k^{-1}$ either has the form

$$\left(\begin{array}{c|c} 1 & \theta^T \\ \hline -\eta_{k,1} & \\ 0 & \\ 0 & X \\ 0 & \end{array}\right)$$

or the form

$$\begin{pmatrix} -\eta_{k,1} & * & * & * & * & * \\ 1 & & & & & \\ \vdots & & & & & \\ 0 & & & X & & \\ 0 & & & & & \\ \vdots & & & & & \\ 0 & & & & & \end{pmatrix}$$

then either

$$V_{k+1}\underset{\sim}{e}_1 = V_k\underset{\sim}{e}_1 - \eta_{k,1}V_k\underset{\sim}{e}_2 \qquad (2-2)$$

or

$$V_{k+1}\underset{\sim}{e}_1 = V_k\underset{\sim}{e}_2 - \eta_{k,1}V_k\underset{\sim}{e}_1 .$$

If pivoting is never done to zero $t_{2,1}^{(k)}$ then equation (2-2) holds for all k and $V_k\underset{\sim}{e}_1$ is linearly independent of the other columns of V_k, and if equation (2-1) is also satisfied, $\eta_{k,1}$ must be approaching 0 and $L_{k,1}$ must be approaching the identity matrix, i.e. the LZ algorithm must be converging. If equation (2-1) holds but the first column of V_k is approaching a multiple of the second column of V_k, then we cannot assert that LZ will converge. It is possible for equation(2-1) to hold without $L_{k,1}$ converging to the identity matrix.

The proof of (2) entails looking at an inverse iteration scheme for finding x such that $(A^T - \rho B^T)\underset{\sim}{x} = \alpha B^T\underset{\sim}{x}$. The iteration scheme can be summarized as follows:

1) Set $\underset{\sim}{x}_0$ to $\underset{\sim}{e}_n$, k to 0.

2) Find $\underset{\sim}{z}$ such that $(A^T - \rho B^T)\underset{\sim}{z} = B^T\underset{\sim}{x}_k$.

3) If $\underset{\sim}{z}$ is large, stop.

4) Set $\underset{\sim}{x}_k$ to $\underset{\sim}{z}/\|\underset{\sim}{z}\|$, k to k+1, go to 2.

If $|d_n| < |d_{n-1}|$ and $\underset{\sim}{e}_n$ is not orthogonal to the left eigenvector of $B(A - \rho B)^{-1}$ corresponding to $1/d_n$, then the inverse iteration scheme will converge and $\underset{\sim}{z}$ will be that eigenvector. The vector $\underset{\sim}{x}_k$ is always in the direction $(A^T - \rho B^T)^{-1} B^T \underset{\sim}{e}_n$, which, according to Corollary 4.1, is also the direction of $V_k^{-T} \underset{\sim}{e}_n$. Hence if the inverse iteration scheme converges, then for large k we may write

$$V_{k+1}^{-T} \underset{\sim}{e}_n = c_k V_k^{-T} \underset{\sim}{e}_n + \underset{\sim}{f}_k \qquad (2-3)$$

where c_k is some scalar and $\underset{\sim}{f}_k$ approaches $\underset{\sim}{\theta}$ as k approaches infinity.

Since $V_{k+1}^{-T} = V_k^{-T} \bar{L}_k^T$

and $\bar{L}_k^T = L_{k,1}^T L_{k,2}^T \cdots\cdots L_{k,n-1}^T$ where $L_{k,i}$ is in $\mathcal{L}_i$

$$= \begin{pmatrix} X & \underset{\sim}{\theta} \\ \underset{\sim}{\theta}^T & 1 \end{pmatrix} \begin{pmatrix} L_{k,n-1}^T \end{pmatrix} \text{ where } X \text{ is a dense matrix}$$

then either

$$V_{k+1}^{-T} \underset{\sim}{e}_n = V_k^{-T} \underset{\sim}{e}_n + \eta_{k,n-1} \sum_{j=1}^{n-2} c_j V_k^{-T} \underset{\sim}{e}_j \qquad (2-4)$$

or

$$V_{k+1}^{-T} \underset{\sim}{e}_n = \sum_{j=1}^{n-2} c_j V_k^{-T} \underset{\sim}{e}_j + \eta_{k,n-1} V_k^{-T} \underset{\sim}{e}_n$$

where the c_j's are products of the multipliers involved in the iteration step.

If pivoting is never done to zero $t_{n,n-1}^{(k)}$, then equation (2-4) holds for all k and the last column of V_k^{-T} must be linearly independent of the other columns of V_k^{-T}. In this case if equation (2-3) is satisfied, then $\eta_{k,n-1}$ must be approaching zero and LZ must be converging. If

equation(2-3) is satisfied but the last column of V_k^{-T} is approaching a linear combination of the other columns of V_k^{-T} as fast as it is converging to an eigenvector then we cannot assert that LZ will converge.

The proofs of (3) and (4) closely resemble those of (1) and (2). For (3) we consider the inverse iteration method for finding x such that $B\underset{\sim}{x} = d(A - \rho B)\underset{\sim}{x}$ and use corollary 5.1. For (4) we consider the power method for determining x such that $B^T(A^T - \rho B^T)^{-1}\underset{\sim}{x} = d\underset{\sim}{x}$ and use corollary 6.1. ∎

Theorem 9 requires more elaboration. It would seem that because the matrices V_k and W_k have determinant 1 for all k, the linear dependence of the columns of these matrices and their inverses should not be an issue. However, if we look at the singular values of these matrices we can get a different picture of the situation. It is possible for these matrices to become singular at the same rate as one of their columns is approaching an eigenvector. If the last two columns of V_k^{-T} are approaching the same vector, then we cannot assume that pivoting will stop and the algorithm will converge. This fact is brought out by the following 2 by 2 example.

Let B be the identity matrix and A the real matrix

$$\begin{bmatrix} a & b \\ b & 0 \end{bmatrix}$$

where $|b| > |a| > 0$. If LZ is applied to the above problem with

a_{22}/b_{22} as the shift policy, the shift is always 0, and A and B are left unchanged by each iteration and the algorithm does not converge. The least singular value of V_k^{-T} is given by the smallest eigenvalue value of

$$\begin{pmatrix} 1 & -a/b \\ -a/b & 1 + \frac{a^2}{b^2} \end{pmatrix}^k$$

which is t^k where

$$t = \frac{2 + \frac{a^2}{b^2} - \left|\frac{a}{b}\right| \sqrt{2 + \frac{a^2}{b^2}}}{2}$$

Since $0 < |a/b| < 1$, $|t| < 1$, which means that as k increases, V_k^{-T} approaches a singular matrix and hence its columns become multiples of each other.

It is ironic that we seem to require that $\underset{\sim}{e}_1$ be nonorthogonal to a right eigenvector of a UHM and that $\underset{\sim}{e}_n$ be nonorthogonal to a left eigenvector of a UHM. If the situation were reversed, there would be no problem, for if $\underset{\sim}{e}_n$ were perpendicular to a right eigenvector of a UHM then the eigenvector must be $\underset{\sim}{\theta}$.

The last interesting fact about Theorem 9 is that it suggests pivoting schemes which should converge although perhaps very slowly. If we find that pivoting is necessary to stably zero $t_{2,1}$ or $b_{2,1}$, we could change the shift so that pivoting is no longer necessary. We could then view the matrices as representing a new problem, and hopefully the two new largest eigenvalues do not have the same modulus.

V. COUNTEREXAMPLES

This section contains a partial listing of 3×3 examples for which the IZ algorithm will not converge when the shift is the eigenvalue of the lowest 2×2 subproblem of $A\underset{\sim}{x} = \lambda B\underset{\sim}{x}$ closest to a_{nn}/b_{nn}. The counterexamples share several characteristics:

1) The subdiagonal elements of A and the matrix B repeat themselves every third iteration.

2) Row and column pivoting are necessary at each stage of the process.

3) The shift is the same for each iteration.

The first property guarantees nonconvergence. The second property is a necessary condition for the first property: if at some stage pivoting is not required to maintain numerical stability, then cycling will not continue as before. The third property indicates that the constant shifting hypothesis of Theorem 8 is realistic in terms of actual computation. In fact, a constant shift may be useful in practice as a warning of nonconvergence. It is significant that no condition is specified for the eigenvalues of a problem. There are counterexamples in which all the eigenvalues are of distinct modulus.

The first class of examples is the basic one. In this case the matrices initially look like

$$\overset{A}{\begin{pmatrix} a & b & c \\ h & 0 & 0 \\ 0 & f & 0 \end{pmatrix}} \qquad \overset{B}{\begin{pmatrix} g & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & m \end{pmatrix}}$$

where the following conditions hold:

$$|a| < |h| \qquad |as/g| < |f| \qquad |am/g| < |c|$$

$$|as/h| < |g| \qquad |am/g| < |f| \qquad |a\ | < |c|$$

$$|b| < |f| \qquad |bmh/(gf)| < |c| \qquad |bm/s| < |c|$$

$$|bm/f| < |g| \qquad |bmh/(cf)| < |c| \qquad |b\ | < |c|.$$

If B is the identity matrix and A is the matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 0 & 0 \\ 0 & 5 & 0 \end{pmatrix}$$

then the above conditions are satisfied.

For problems in class 1 the shift is zero for each iteration. After one iteration the matrices are

$$\overset{A}{\begin{pmatrix} as/g & bhm/(fg) & h \\ f & 0 & 0 \\ 0 & c & 0 \end{pmatrix}} \qquad \overset{B}{\begin{pmatrix} s & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & g \end{pmatrix}}.$$

After two iterations the matrices look like

$$\overset{A}{\begin{pmatrix} am/g & bhm/(cs) & f \\ c & 0 & 0 \\ 0 & h & 0 \end{pmatrix}} \qquad \overset{B}{\begin{pmatrix} m & 0 & 0 \\ 0 & g & 0 \\ 0 & 0 & s \end{pmatrix}}.$$

After three iterations the matrices return their original forms.

The second class of examples consists of those problems which fall into the first class after one iteration but initially look like

$$\overset{A}{\begin{pmatrix} x & y & f \\ c & d & 0 \\ 0 & h & 0 \end{pmatrix}} \qquad \overset{B}{\begin{pmatrix} m & p & q \\ 0 & g & 0 \\ 0 & 0 & s \end{pmatrix}}$$

where

$$|x| < |c| \qquad |y - \frac{x}{c} d| < |h|$$

$$|p - \frac{x}{c} g| < |m| \qquad | q - (y - \frac{x}{c} d)s/h| < |m|.$$

The first shift is 0 and after one iteration the matrices of this class have the form

$$\overset{A}{\begin{pmatrix} a & b & c \\ h & 0 & 0 \\ 0 & f & 0 \end{pmatrix}} \qquad \overset{B}{\begin{pmatrix} g & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & m \end{pmatrix}}$$

where a and b are complicated expressions.

The third class of counterexamples includes problems in which A and B are initially given by

$$\overset{A}{\begin{pmatrix} a & b & c \\ h & d & 0 \\ 0 & f & d \end{pmatrix}} \qquad \overset{B}{\begin{pmatrix} m & f & q \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}}.$$

The shift is d/m and after shifting, a_{22} and a_{33} are zero. If this new problem is in class 2, then in one iteration before the shift has been added back A and B would look like

$$\begin{pmatrix} x & y & h \\ f & 0 & 0 \\ 0 & c & 0 \end{pmatrix} \qquad \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}$$

where x and y are again nonsimple expressions. Shifting back sets a_{22} and a_{33} to d. From here it should be clear that the matrices will repeat themselves every third iteration if their elements satisfy the appropriate magnitude relationships.

If in the previous example, a_{22} were initially zero, the first shift would again be d/m . Performing one iteration and shifting back set the element a_{22} to d and revert us to the previous example.

These counterexamples indicate that assuming the algorithm uses a constant shift is reasonable and that given this assumption, the structure of L and M is important. The above examples were all constructed by assuming that pivoting was necessary to maintain stability at every step of the algorithm. Indeed if pivoting ceases at some stage, then "cycling" will not continue as before. It is doubtful that without an analysis of a given shift strategy we can weaken significantly the criteria given earlier for global convergence. Moreover, it is extremely doubtful that such an analysis could give us more than assymptotic convergence results.

APPENDIX

NUMERICAL RESULTS AND FORTRAN PROGRAM

The algorithm described in Chapter 1 has been implemented in a Fortran program. The program is designed to find the eigensystem for complex matrices, and consists of two subroutines which must be called separately. The first subroutine,GELHES, reduces A to upper Hessenberg form and B to upper triangular form. If A and B are already in these forms and no eigenvectors are required, then calling GELHES is unnecessary. The second subroutine GLR finds the eigenvalues and, if requested, the eigenvectors of the system. The parameters involved in the subroutine calls are described in the comments at the beginning of each subroutine. Both subroutines use the subroutine RABS to compute the norm of a complex number.

It should be emphasized that the variable MACHEPS is machine dependent. It is β^{1-t} where the machine gives t digits in base β. It is set for the IBM 360 double precision mode.

Our program has been finding eigenvalues which correspond to problems close to the given problems: our relative residuals have always been close to the precision of the machine.

For our test examples the total number of iterations has been roughly 3 times the order of the matrices. In general, for a matrix of order n, the time required on the IBM 360 model 67 in Fortran H,opt=2, has been about $.75n^3$ milliseconds if eigenvectors are computed and $.4n^3$ milliseconds if they are not.

The example given below were generated using integer arithmetic by multiplying two bidiagonal matrices by random nonsingular transformations. The problems were run on the IBM 360 Fortran G compiler. The relative residual is the quantity

$$\frac{|\beta_i A\underset{\sim}{x}_i - \alpha_i B\underset{\sim}{x}_i|_\infty}{|\beta_i|\,||A||_\infty + |\alpha_i|\,||B||_\infty}$$

where α_i and β_i are the ith diagonal elements of the triangularized A and B and $\underset{\sim}{x}_i$ is the ith eigenvector.

In the first example A has rank 4 and B has rank 3 and their null spaces intersect. Since the rank of B is less than the rank of A, there is an eigenvalue which might be regarded as infinite because a small perturbation in B would yield a large eigenvalue. Indeed an eigenvalue of 10^{16} was found. The problem is also "ill -disposed", because for any vector $\underset{\sim}{x}$ in the intersection of the null spaces, any scalar λ will satisfy $A\underset{\sim}{x} = \lambda B\underset{\sim}{x}$, and may be considered an eigenvalue of the problem. The example also has three genuine, finite eigenvalues which the algorithm was able to find accurately up to the precision of the machine despite the presence of the two spurious eigenvalues.

In the second example there are two double roots. The first corresponds to a quadratic elementary divisor and, as expected, is accurate only up to the square root of the machine precision. The second corresponds to two linear elementary divisors and is accurate almost up to the machine precision.

Example 1

THE MATRIX A:

575.+	-116.I	205.+	-460.I	-30.+	456.I	40.+	-352.I	-165.+	-332.I
355.+	-356.I	805.+	-324.I	-670.+	184.I	780.+	-200.I	235.+	100.I
345.+	-103.I	590.+	-440.I	-400.+	352.I	625.+	-412.I	-100.+	-16.I
210.+	-200.I	-215.+	208.I	190.+	-292.I	-115.+	258.I	115.+	24.I
-545.+	156.I	-115.+	282.I	-30.+	-276.I	-40.+	222.I	115.+	248.I

THE MATRIX B:

69.+	-387.I	88.+	504.I	-68.+	-612.I	143.+	567.I	36.+	468.I
241.+	9.I	48.+	108.I	28.+	36.I	139.+	63.I	-76.+	-468.I
219.+	-333.I	70.+	558.I	-8.+	-504.I	167.+	567.I	-24.+	-72.I
46.+	414.I	96.+	-675.I	-56.+	630.I	130.+	-693.I	-52.+	18.I
-91.+	189.I	-112.+	-207.I	76.+	306.I	-177.+	-252.I	-8.+	-378.I

TRUE EIGENVALUE

1	INFINITE EIGENVALUE	
2	-0.500000000000000D 00+	0.500000000000000D 00 I
3	-0.500000000000000D 00+	-0.500000000000000D 00 I
4	0.166666666666667D 01+	-0.666666666666667D 00 I
5	ANY SCALAR	

ALPHA

1	-0.183210649328370D 03+	-0.206212656356070D 03 I
2	-0.139921151742898D 03+	-0.113785397263304D 01 I
3	-0.642373476326618D 03+	-0.371434242540361D 03 I
4	0.143000000000000D 03+	-0.701000000000000D 03 I
5	-0.146368201958883D-12+	0.571615032881263D-14 I

BETA

1	-0.575137927367557D-13+	-0.778836003625250D-14 I
2	0.139783297770265D 03+	0.141059005715531D 03 I
3	0.101330771886698D 04+	-0.270039233786250D 03 I
4	0.210000000000000D 03+	-0.333000000000000D 03 I
5	-0.157644233166012D-12+	-0.220174709424694D-13 I

COMPUTED EIGENVALUE

1	0.360495505805896D 16+	0.399724254812665D 16 I
2	-0.500000000000001D 00+	0.500000000000000D 00 I
3	-0.500000000000000D 00+	-0.500000000000000D 00 I
4	0.166666666666667D 01+	-0.666666666666666D 00 I
5	0.995739603559031D 00+	-0.162760495914164D 00 I

	RELATIVE ERROR	RELATIVE RESIDUAL	NO. OF ITERATIONS
1	0.999994922874991D 00	0.405556314396225D-16	0
2	0.457906997657277D-15	0.961773509713645D-16	0
3	0.471244785465692D-15	0.112430158655119D-15	1
4	0.457906997657277D-15	0.107364026307787D-15	7
5	0.0	0.920015101641430D-16	0

Example 2

```
NONLINEAR DIVISOR
 THE  MATRIX A:

  311.+ -339.|   397.+ -903.|    88.+-1032.|  -250.+  240.|  -112.+ -672.|
   64.+ -278.| -1053.+  138.|   127.+ -694.|   888.+-1000.|   183.+ -808.|
 -124.+  429.|  -858.+ 1023.|    81.+  176.|   679.+ -564.|    -3.+ 1124.|
   99.+ -776.|   531.+   11.|   115.+ -139.|  -377.+  848.|   -74.+   44.|
 -125.+ -709.|  -560.+ 1057.|   153.+  164.|   458.+   52.|   135.+  212.|

THE MATRIX B:

  180.+   11.|   360.+   47.|    72.+  104.|  -180.+    0.|  -144.+   96.|
  168.+    6.|  -546.+   60.|   210.+   46.|   588.+   24.|   -42.+   96.|
   90.+  -61.|  -450.+  -47.|   138.+  -40.|   450.+   -4.|    -6.+ -124.|
   42.+   80.|   336.+  -51.|   -24.+   19.|  -258.+  -56.|   -60.+    4.|
   12.+   77.|  -354.+  -65.|    66.+  -28.|   312.+  -52.|    30.+  -28.|
```

Reproduced from best available copy.

```
   TRUE EIGENVALUE
  1    0.116666666666667D 01+  -0.133333333333333D 01 |
  2    0.116666666666667D 01+  -0.133333333333333D 01 |
  3   -0.900000000000000D 01+   0.100000000000000D 01 |
  4   -0.900000000000000D 01+  -0.100000000000000D 01 |
  5   -0.900000000000000D 01+  -0.100000000000000D 01 |
ALPHA
  1   -0.220225259386724D 04+   0.253104871789474D 04 |
  2   -0.670083541237466D 03+   0.729094818864090D 03 |
  3   -0.422890641115618D 03+   0.158688091695233D 04 |
  4    0.130967144237819D 03+   0.587947693142187D 03 |
  5   -0.914681533504013D 01+   0.715031183306420D 02 |
BETA
  1   -0.189367208803889D 04+   0.527363038922533D 01 |
  2   -0.558761597145862D 03+  -0.136462582493453D 02 |
  3    0.657571547194251D 02+  -0.169913751359212D 03 |
  4   -0.270323413571044D 02+  -0.623239279761203D 02 |
  5    0.131929508359252D 00+  -0.795944975988893D 01 |
   COMPUTED EIGENVALUE
  1    0.116666666583722D 01+  -0.133333335008736D 01 |
  2    0.116666666749611D 01+  -0.133333331657931D 01 |
  3   -0.900000000000000D 01+   0.999999999999980D 00 |
  4   -0.900000000000000D 01+  -0.999999999999997D 00 |
  5   -0.900000000000000D 01+  -0.999999999999978D 00 |
          RELATIVE ERROR          RELATIVE RESIDUAL        NO. OF ITERATIONS
  1    0.703338756125049D-08   0.530711046259343D-16   0
  2    0.703338729479697D-08   0.355091199667428D-16   1
  3    0.201227923213310D-14   0.164571539197435D-15   3
  4    0.539845945723982D-15   0.127436297695855D-15   1
  5    0.222044604925031D-14   0.801946522734460D-16   1
```

```
C
C
      SUBROUTINE GELHES(ND,N,A,B,WANTX,X,EPSA,EPSB)
C
C     THIS SUBROUTINE REDUCES THE COMPLEX MATRIX A TO UPPER
C     HESSENBERG FORM AND REDUCES THE COMPLEX MATRIX B TO
C     TRIANGULAR FORM
C
C     INPUT PARAMETERS:
C
C     ND  THE ROW DIMENSION OF THE MATRICES A,B,X
C
C     N   THE ORDER OF THE PROBLEM
C
C     A   A COMPLEX MATRIX
C
C     B   A COMPLEX MATRIX
C
C     WANTX A LOGICAL VARIABLE WHICH IS SET TO .TRUE. IF
C           THE EIGENVECTORS ARE WANTED. OTHERWISE IT SHOULD
C         BE SET TO .FALSE.
C
C     OUTPUT PARAMETERS:
C
C     A  ON OUTPUT A IS AN UPPER HESSENBERG MATRIX, THE
C        ORIGINAL MATRIX HAS BEEN DESTROYED
C
C     B  AN UPPER TRIANGULAR MATRIX, THE ORIGINAL MATRIX
C        HAS BEEN DESTROYED
C
C     X  CONTAINS THE TRANSFORMATIONS NEEDED TO COMPUTE
C        THE EIGENVECTORS OF THE ORIGINAL SYSTEM
C
C     EPSA  THE NORM OF A*THE PRECISION OF THE MACHINE
C
C     EPSB  THE NORM OF B*THE PRECISION OF THE MACHINE
C
C**** THE VALUE OF MACHEP IS MACHINE DEPENDENT*******
C*****IT IS SET FOR THE IBM 360 MACHINE, DOUBLE PRECISION*****
C
C      PROBLEMS WITH THIS SUBROUTINE SHOULD BE DIRECTED TO:
C
C          LINDA KAUFMAN
C          SERRA HOUSE
C          COMPUTER SCIENCE DEPARTMENT
C          STANFORD UNIVERSITY
C
      COMPLEX *16 Y,A(ND,ND),B(ND,ND),X(ND,ND)
      REAL*8 ANI,BNI,C,RABS,D,EPSA,EPSB,MACHEP,ANORM,BNORM
      LOGICAL WANTX
      NM1=N-1
C
C   COMPUTE EPSA,EPSB
```

```
C
      MACHEP=2.22D-16
      ANORM = 0.
      BNORM = 0.
      DO    5 I=1,N
         ANI = 0.
         IF (I.NE.1) ANI = RABS(A(I,I-1))
         BNI = 0.
         DO 3   J=I,N
            ANI = ANI + RABS(A(I,J))
            BNI = BNI + RABS(B(I,J))
3        CONTINUE
         IF (ANI.GT.ANORM) ANORM = ANI
         IF (BNI.GT.BNORM) BNORM = BNI
5     CONTINUE
      IF (ANORM.EQ.0.) ANORM = MACHEP
      IF (BNORM.EQ.0.) BNORM = MACHEP
      EPSA = MACHEP*ANORM
      EPSB = MACHEP*BNORM
C
C     REDUCE B TO TRIANGULAR FORM USING ELEMENTARY TRANSFORMATIONS
C
      DO 30 I=1,NM1
      D=0.0D0
           IP1=I+1
           DO 10 K=IP1,N
                C=RABS(B(K,I))
                IF (C.LE.D) GO TO 10
                     D=C
                     II=K
10         CONTINUE
           IF (D.EQ.0.D0) GO TO 30
           IF (D.LE.RABS(B(I,I))) GO TO 15
           DO 11  J=1,N
                Y=A(I,J)
                A(I,J)=A(II,J)
11              A(II,J)=Y
           DO 12 J=I,N
                Y=B(I,J)
                B(I,J)=B(II,J)
12              B(II,J)=Y
15         DO 20 J=IP1,N
                Y=B(J,I)/B(I,I)
                IF (RABS(Y).EQ.0.D0) GO TO 20
                DO 18 K=1,N
18                   A(J,K)=A(J,K)-Y*A(I,K)
                DO 19 K=IP1,N
19                   B(J,K)=B(J,K)-Y*B(I,K)
                B(J,I)=(0.D0,0.D0)
20         CONTINUE
30    CONTINUE
C
C INITIALIZE X
C
      IF (.NOT.WANTX) GO TO 40
```

```
      DO 38 I=1,N
          DO 37 J=1,N
37            X(I,J)=(0.D0,0.D0)
38        X(I,I)=(1.D0,0.0D0)
C
C     REDUCE A TO UPPER HESSENBERG FORM
C
40    NM2=N-2
      IF (NM2.LT.1) GO TO 100
      DO 90 J=1,NM2
          JM2=NM1-J
          JP1=J+1
              DO 80 II=1,JM2
               I=N+1-II
               IM1=I-1
               IF (RABS(A(I,J)).LE.RABS(A(IM1,J))) GO TO 50
               DO 45 K=J,N
                    Y=A(I,K)
                    A(I,K)=A(IM1,K)
45                  A(IM1,K)=Y
               DO 46 K=IM1,N
                    Y=B(I,K)
                    B(I,K)=B(IM1,K)
46                  B(IM1,K)=Y
50             IF (RABS(A(I,J)).EQ.0.D0) GO TO 58
               Y=A(I,J)/A(IM1,J)
               DO 52 K=JP1,N
52                  A(I,K)=A(I,K)-Y*A(IM1,K)
               A(I,J)=(0.0D0,0.D0)
               DO 54 K=IM1,N
54                  B(I,K)=B(I,K)-Y*B(IM1,K)
C     TRANSFORMATION FROM THE RIGHT
58             IF (RABS(B(I,IM1)).LE.RABS(B(I,I))) GO TO 70
               DO 60 K=1,I
                    Y=B(K,I)
                    B(K,I)=B(K,IM1)
60                  B(K,IM1)=Y
               DO 64 K=1,N
                    Y=A(K,I)
                    A(K,I)=A(K,IM1)
64                  A(K,IM1)=Y
               IF (.NOT.WANTX) GO TO 70
               DO 68 K=1,N
                    Y=X(K,I)
                    X(K,I)=X(K,IM1)
68                  X(K,IM1)=Y
70             IF (RABS(B(I,IM1)).EQ.0.D0) GO TO 80
               Y=B(I,IM1)/B(I,I)
               DO 72 K=1,IM1
72                  B(K,IM1)=B(K,IM1)-Y*B(K,I)
               B(I,IM1)=(0.D0,0.D0)
               DO 74 K=1,N
74                  A(K,IM1)=A(K,IM1)-Y*A(K,I)
               IF (.NOT.WANTX) GO TO 80
               DO 76 K=1,N
```

```
76                    X(K,IM1)=X(K,IM1)-Y*X(K,I)
80          CONTINUE
90     CONTINUE
100    RETURN
       END

      SUBROUTINE GLR(ND,NN,A,B,*,X,ITER,EPSA,EPSB,WANTX,EIGA,EIGB)
C
C THIS SUBROUTINE SOLVES THE GENERALIZED EIGENVALUE PROBLEM
C                A X  = LAMBDA B X
C WHERE A IS A COMPLEX UPPER HESSENBERG MATRIX OF ORDER NN AND B IS
C A COMPLEX UPPER TRIANGULAR MATRIX OF ORDER NN
C
C
C INPUT PARAMETERS
C
C
C ND  ROW DIMENSION OF THE MATRICES A,B,X,ITER,EIGA,EIGB
C
C NN   ORDER OF THE PROBLEM
C
C A    AN NN X NN UPPER HESSENBERG COMPLEX MATRIX
C
C B    AN NN X NN UPPER TRIANGULAR COMPLEX MATRIX
C
C * ERROR RETURN, IF AFTER 30 ITERATIONS, THE NORM OF THE
C     SUBDIAGONAL OF A HAS NOT SHOWN A SUFFICIENT DECREASE
C
C X   CONTAINS TRANSFORMATIONS TO OBTAIN EIGENVECTORS OF
C     ORIGINAL SYSTEM
C     IF GELHES HAS NOT BEEN USED, X SHOULD BE THE IDENTITY MATRIX
C
C WANTX LOGICAL VARIABLE WHICH SHOULD BE SET TO .TRUE. IF EIGENVECTORS
C     ARE WANTED. OTHERWISE IT SHOULD BE SET TO FALSE
C
C EPSA    THE NORM OF A TIMES THE MACHINE PRECISION. NEED NOT BE
C         SET IF GELHES HAS BEEN USED
C EPSB    THE NORM OF B TIMES THE MACHINE PRECISION. NEED NOT
C         BE SET IF GELHES HAS BEEN USED
C
C
C OUTPUT PARAMETERS
C
C
C W      THE ITH COLUMN CONTAINS THE ITH EIGENVECTOR IF EIGENVECTORS ARE
C        REQUESTED
C
C ITER AN INTEGER ARRAY OF LENGTH NN WHOSE ITH ENTRY CONTAINS THE NUMBER
C         OF ITERATIONS NEEDED TO FIND THE ITH EIGENVALUE
C
C EIGA    AN NN ARRAY CONTAINING THE DIAGONAL OF A
C
C EIGB   AN NN ARRAY CONTAINING THE DIAGONAL OF B
C
C THE ITH EIGENVALUE CAN BE FOUND BY DIVIDING EIGA(I) BY EIGB(I)
C WATCH OUT FOR EIGB(I) BEING ZERO
C
C******** THE QUANTITY MACHEP IS MACHINE DEPENDENT*********
C********* IT IS SET FOR THE IBM 360, DOUBLE PRECISION********
```

```
C
C PROBLEMS WITH THIS SUBROUTINE SHOULD BE DIRECTED TO
C          LINDA KAUFMAN
C          SERRA HOUSE, COMPUTER SCIENCE DEPARTMENT
C          STANFORD UNIVERSITY
C
C
      COMPLEX*16 A(ND,ND),B(ND,ND),EIGA(ND),EIGB(ND)
      COMPLEX*16 S,T,W,Y,Z,DCMPLX,CDSQRT,TS
      INTEGER  ITER(ND)
      COMPLEX*16 ALFM,BETM,D,SL,DEN,ANN,ANM1N,ANM1M1
      REAL*8 EPSA,EPSB,SS,R,OLD,NEW
      COMPLEX*16 X(ND,ND)
      REAL*8 MACHEP/2.22D-16/,D0,D1,D2,E0,E1,RABS,DABS
      LOGICAL WANTX
      N=NN
C
      IF (N.LE.1) GO TO 100
10    ITS=0
      NM1=N-1
11    DO 12 LB=2,N
         L=N+2-LB
         IF(RABS(A(L,L-1)).LE.MACHEP*(RABS(A(L-1,L-1))
     *                                  +RABS(A(L,L)))) GO TO 13
12    CONTINUE
      L=1
13    IF(L.EQ.N) GO TO 100
      IF (ITS.LT.30) GO TO 20
      IF (ITS.GT.30) GO TO 16
           OLD=0.D0
           DO 15 I=1,NM1
15            OLD=OLD+RABS(A(I+1,I))
           GO TO 20
16         NEW=0.D0
           DO 19 I=1,NM1
19            NEW=NEW+RABS(A(I+1,I))
           IF (NEW.GT.0.5*OLD) RETURN 1
           OLD=NEW
C
C CHECK FOR 2 CONSECUTIVE SMALL SUBDIAGONAL ELEMENTS
C
20    IF(N.EQ.L+1) GO TO 25
      D2=RABS(A(N-1,N-1))
      E1=RABS(A(N,N-1))
      D1=RABS(A(N,N))
      NL=N-(L+1)
      DO 24 MB=1,NL
         M=N-MB
         E0=E1
         E1=RABS(A(M,M-1))
         D0=D1
         D1=D2
         D2=RABS(A(M-1,M-1))
         IF(E0*E1.LE.MACHEP*D1*(D0+D1+D2)) GO TO 26
24    CONTINUE
```

```
25    M=L
26    CONTINUE
C
      IF(ITS.EQ.10.OR.ITS.EQ.20) GO TO 38
C
C COMPUTE SHIFT AS EIGENVALUE OF LOWER 2 BY 2
C
      ANN=A(N,N)
      ANM1N=A(NM1,N)
      ANM1M1=A(NM1,NM1)
      S=ANN*B(NM1,NM1)-(A(N,NM1))*B(NM1,N)
      W=A(N,NM1)*B(N,N)*(ANM1N*B(NM1,NM1)-
     1 B(NM1,N)*ANM1M1)
      Y=(ANM1M1*B(N,N)-S)/2.
      Z=CDSQRT(Y*Y+W)
      IF (RABS(Z).EQ.0.D0) GO TO 36
      D0=Y/Z
      IF(D0.LT.0.D0) Z=-Z
36    DEN=(Y+Z)*B(NM1,NM1)*B(N,N)
      W=A(M,M)*DEN-B(M,M)*((Y+Z)*S-W)
      Z=A(M+1,M)*DEN
      GO TO 40
C
C AD-HOC SHIFT
C
38    W=A(N,N-1)
      Y=A(N-1,N-2)
      W=A(M,M)-DCMPLX(RABS(W),RABS(Y))*B(M,M)
      Z=A(M+1,M)
40    CONTINUE
      ITS=ITS+1
C
C FIND L AND M AND SET A=LAM AND B=LBM
C
       NP1=N+1
       LUR1=L
       NNORN=N
       IF (.NOT.WANTX) GO TO 42
           LUR1=1
           NNURN=NN
42     DO 9C I=M,NM1
           J=I+1
C
C FIND ROW TRANSFORMATIONS TO RESTORE A TO
C UPPER HESSENBERG FORM. APPLY TRANSFORMATIONS
C TO A AND B
C
           IF (I.EQ.M) GO TO 50
           W=A(I,I-1)
           Z=A(J,I-1)
50         IF (RABS(W).GE.RABS(Z)) GO TO 60
C
C MUST PIVOT
C
           DU 55 K=I,NNORN
```

```
                Y=A(I,K)
                A(I,K)=A(J,K)
                A(J,K)=Y
                Y=B(I,K)
                B(I,K)=B(J,K)
55              B(J,K)=Y
             Y=W/Z
             IF (I.GT.M) A(I,I-1)=A(J,I-1)
             GO TO 62
60           Y=Z/W
62           IF(RABS(Y).EQ.0.D0) GO TO 65
             DO 64 K=I,NNORN
                A(J,K)=A(J,K)-Y*A(I,K)
64              B(J,K)=B(J,K)-Y*B(I,K)
             IF (I.GT.M) A(J,I-1)=(0.D0,0.D0)
C
C PERFORM TRANSFORMATIONS FROM RIGHT TO RESTORE B TO
C   TRIANGLULAR FORM
C APPLY TRANSFORMATIONS TO A
C
65           IF (RABS(B(J,I)).EQ.0.D0) GO TO 11
             IF (RABS(B(J,J)).GE.RABS(B(J,I))) GO TO 80
C
C MUST PIVOT COLUMNS
C
             DO 70 K=LOR1,J
                Y=A(K,J)
                A(K,J)=A(K,I)
                A(K,I)=Y
                Y=B(K,J)
                B(K,J)=B(K,I)
70              B(K,I)=Y
             IF (I.EQ.NM1) GO TO 75
                Y=A(J+1,J)
                A(J+1,J)=A(J+1,I)
                A(J+1,I)=Y
75           IF(.NOT.WANTX) GO TO 80
             DO 78 K=1,NN
                 Y=X(K,J)
                 X(K,J)=X(K,I)
78               X(K,I)=Y
80           IF (RABS(B(J,I)).EQ.0.D0) GO TO 90
             Z=B(J,I)/B(J,J)
             DO 82 K=LOR1,J
                  A(K,I)=A(K,I)-Z*A(K,J)
82                B(K,I)=B(K,I)-Z*B(K,J)
             B(J,I)=(0.D0,0.D0)
             IF (I.LT.NM1) A(I+2,I)=A(I+2,I)-Z*A(I+2,J)
             IF(.NOT.WANTX) GO TO 90
             DO 85 K=1,NN
85                 X(K,I)=X(K,I)-Z*X(K,J)
90        CONTINUE
       GO TO 11
C
100    CONTINUE
```

```
      EIGA(N)=A(N,N)
      EIGB(N)=B(N,N)
      IF (N.EQ.1) GO TO 110
      ITER(N)=ITS
      N=NM1
      IF (N.GT.1) GO TO  10
      ITER(1)=0
      GO TO 100
C
C     FIND EIGENVECTORS USING B FOR INTERMEDIATE STORAGE
C
110    IF(.NOT.WANTX) RETURN
      M=NN
115   CONTINUE
      ALFM=A(M,M)
      BETM=B(M,M)
      B(M,M)=(1.D0,0.D0)
        L = M-1
        IF (L.EQ.0) GO TO 140
120     CONTINUE
           L1 = L+1
           SL = 0.
           DO 130 J=L1,M
              SL = SL + (BETM*A(L,J)-ALFM*B(L,J))*B(J,M)
130        CONTINUE
           D = BETM*A(L,L)-ALFM*B(L,L)
           IF (RABS(D).EQ.0.) D = (EPSA+EPSB)/2.
           B(L,M) = -SL/D
           L = L-1
140     IF (L.GT.0) GO TO 120
        M=M-1
        IF (M.GT.0) GO TO 115
C
C  TRANSFORM TO ORIGINAL COORDINATE SYSTEM
C
      M = NN
200   CONTINUE
        DO 220 I=1,NN
           S = 0.
           DO 210 J=1,M
              S = S + X(I,J)*B(J,M)
210        CONTINUE
           X(I,M) = S
220     CONTINUE
        M = M-1
      IF (M.GT.0) GO TO 200
C
C  NORMALIZE SO THAT LARGEST COMPONENT = 1.
C
      M = NN
230   CONTINUE
        SS = 0.
        DO 235 I=1,NN
           R =  RABS(X(I,M))
           IF (R.LT.SS) GO TO 235
```

```
            SS = R
            D = X(I,M)
235      CONTINUE
         IF (SS.EQ.0.D0) GO TO 245
         DO 240 I=1,NN
            X(I,M) = X(I,M)/D
240      CONTINUE
245   M = M-1
      IF (M.GT.0) GO TO 230
      RETURN
      END


      REAL FUNCTION RABS*8(Z)
      COMPLEX*16 Z,ZZ
      REAL*8 T(2),DABS
      EQUIVALENCE(ZZ,T(1))
      ZZ=Z
      RABS=DABS(T(1))+DABS(T(2))
      RETURN
      END
```

Acknowledgements

I am deeply indebted to my advisor Professor Cleve Moler for his guidance, encouragement, and enthusiasm. I thank him for patiently explaining to me the intricasies of the QZ,QR, and LP algorithms and for challenging me to find the LZ algorithm. I am grateful to Professor George Forsythe, Professor John Herriot,and Professor Gene Golub for reading the manuscript and to Professor Michael Osborne for finding an error in one of the earlier versions of one of the convergence proofs. I would also like to thank Mary Bodley and Fran Brumbaugh for typing the manuscript.

REFERENCES

1. H.J. Buurema, "A Geometric Proof of the Convergence of the QR Method", Report TW-62, Mathematisch Instituut, Groningen, North East Netherlands, 1968.

2. C.R. Crawford, "The Numerical Solution of the Generalized Eigenvalue Problem", University of Michigan Ph.D. Thesis, 1971.

3. G. Fix and R. Heiberger, "An Algorithm for the Ill-conditioned Generalized Eigenvalue Problem", to be published in Num. Math.

4. G. Forsythe and C. Moler, Computer Solution of Linear Algebraic Systems, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1967.

5. J.G. Francis, "The QR Transformation-A Unitary Analogue to the LR Transformation", Computer Journal 4,, 265-271, 332-345, 1961-1962.

6. F.R. Gantmacher, Application of the Theory of Matrices, Intersciences Publishers, Inc., New York, 1959.

7. G.H. Golub, R. Underwood, J.H. Wilkinson, "The Lanczos Algorithm for the Symmetric $Ax=\lambda Bx$ Problem", Stanford Computer Science Report 270, March 1972.

8. P. Lancaster, Lambda Matrices and Vibrating Systems, Pergamon Press, New York, 1966.

9. R.S. Martin and J.H. Wilkinson, "Reduction of the Symmetric Eigenproblem $Ax = \lambda Bx$ and Related Problems to Standard Form", Numer. Math. 11, 99-110, 1968.

10. C.B. Moler and G.W. Stewart, "An Algorithm for the Generalized Matrix Eigenvalue Problem $Ax=\lambda Bx$", Stanford Computer Science Report 232, August 1971.

11. B.N. Parlett, "Global Convergence of the Basic QR Algorithm for Hessenberg Matrices", Math. Comp. 22, 803-817, 1968.

12. -------and W. Kahan, "On the Convegrence of the Practical QR Algorithm", Proceedings of the IFIP Congress 1968, A25-A30., 1968.

13. -------and W.G. Poole, "A Geometric Theory for the QR, LU, and Power Iterations", Berkeley Computer Science Technical Report, 1971.

14. B.N. Parlett,"Canonical Decomposition of Hessenberg Matrices",Math Comp. 21,223-227,1967.

15. G.Peters and J.H. Wilkinson,"Ax =λBx and the Generalized Eigenproblem",Siam. J. of Numer. Anal. 7,479-492,1970.

16. G. Peters and J.H. Wilkinson,"Eigenvectors of real and complex matrices by LR and QR triangularizations", Numer. Math. 16,191-204,1970.

17. H. Rutishauser, "Solution of the Eigenvalue Problem with the LR Transformation", National Bureau of Standards Applied Math Series 49, January 1958.

18. G.W.Stewart,"On the Sensitivity of the Eigenvalue Problem Ax= λBx", Center of Numerical Analysis, University of Texas Report 13, March 1971.

19. J.H.Wilkinson,The Algebraic Eigenvalue Problem Oxford University Press,1965.